

JSP (Java Sever Page)

- 웹 프로그래밍 시작하기

01. 네트워크, 인터넷, 웹

■ 웹 서비스의 동작 과정

■ 웹 서버 소프트웨어

- 서버에서 웹 서비스를 제공하는 소프트웨어
 - 아파치(Apache)
 - 엔진(Enginx)
 - 마이크로소프트 IIS(Internet Information Server) 가 대표적임.

■ 클라이언트 소프트웨어

- 웹 서비스를 이용하기 위한 클라이언트 소프트웨어 → 웹 브라우저(Web Browser)
 - 크롬(Chrome)
 - 파이어폭스(Firefox)
 - 애플 사파리(Safari)
 - 인터넷 익스플로러(Internet Explorer) 등

01. 웹 서비스

: /

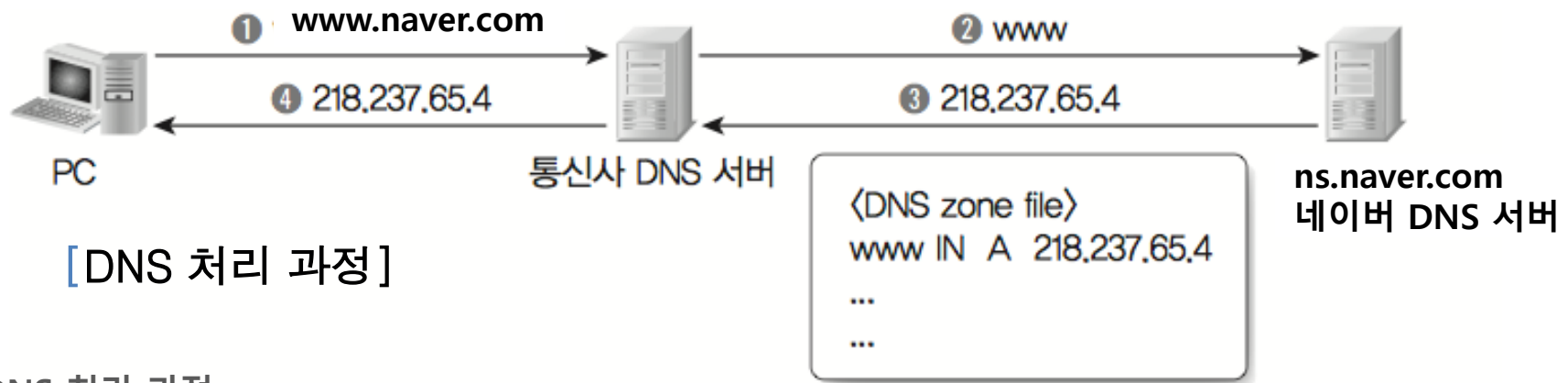
ip address

127.0.0.1

■ 도메인 네임 시스템

▪ DNS(Domain Name System)

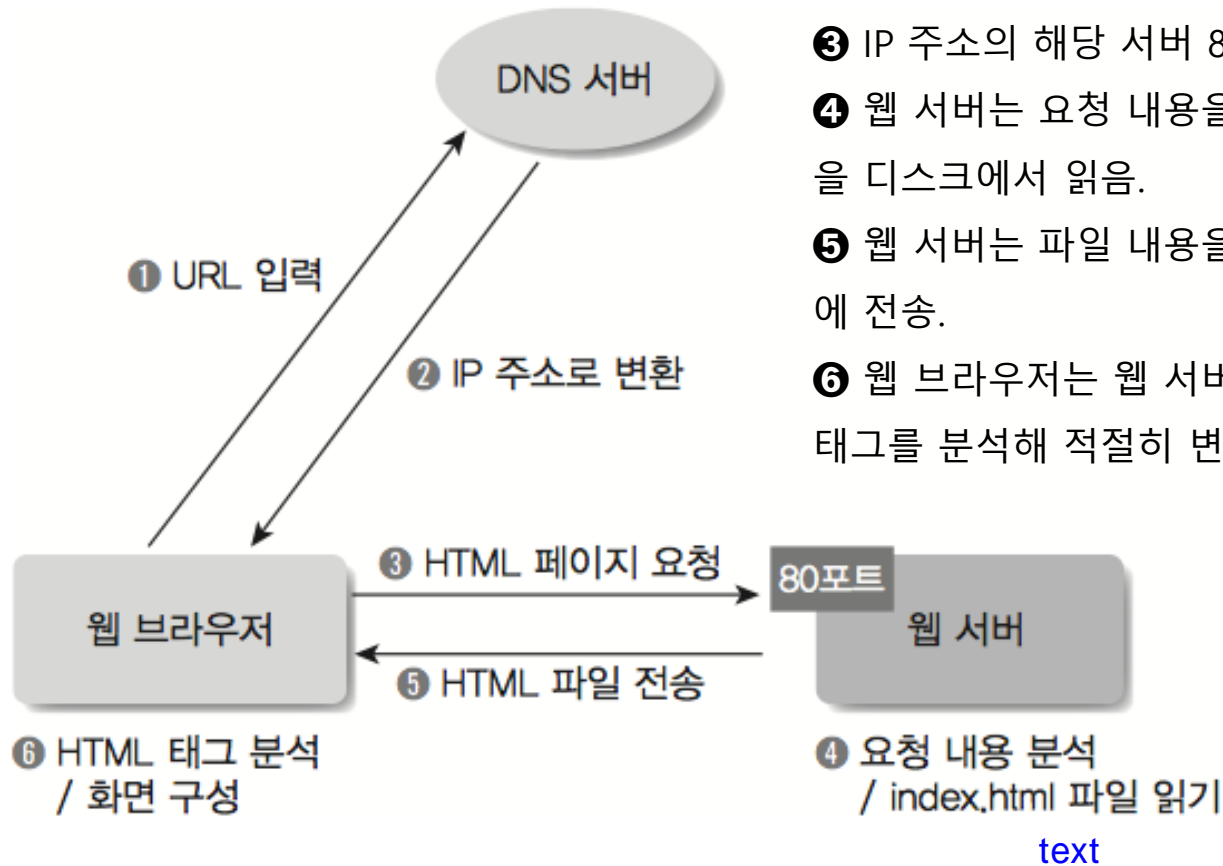
- 인터넷에 연결된 컴퓨터 구분을 위해 사용하는 IP주소 대신 알기 쉬운 이름 형태로 컴퓨터를 구분하기 위한 인터넷 서비스/시스템 중 하나.
- 예) www.naver.com, www.daum.net 등.



▪ DNS 처리 과정

- ① 네트워크 서비스 제공 회사(KT, SK브로드밴드, LGU+ 등)의 DNS 서버에 도메인 이름을 요청한다.
- ② DNS 서버는 네이버 DNS 서버인 ns.naver.com에 도메인 이름의 서버 IP 주소를 요청한다.
- ③ 네이버 DNS 서버는 DNS zone file에서 `www.naver.com` 서버 IP 주소를 찾아서 알려준다.
- ④ DNS 서버는 다시 요청한 클라이언트에 IP 주소를 알려준다.

01. 네트워크, 인터넷, 웹



클라이언트와 서버간 동작 과정

- ① 웹 브라우저에서 `http://www.xxx.com/index.html`을 입력
- ② `www.xxx.com` 도메인의 IP 주소를 DNS 서버로부터 받음
- ③ IP 주소의 해당 서버 80번 포트로 접속을 시도
- ④ 웹 서버는 요청 내용을 분석하고 요청된 `index.html` 파일을 디스크에서 읽음.
- ⑤ 웹 서버는 파일 내용을 텍스트 그대로 요청한 클라이언트에 전송.
- ⑥ 웹 브라우저는 웹 서버에서 보내는 텍스트 내용 중 HTML 태그를 분석해 적절히 변환하여 화면을 구성.

02. 웹 프로그래밍 언어와 주요 기술

1. 웹 프로그램의 개요

- 별도의 설치 없이 서버에 접속하는 것만으로 필요한 기능/서비스를 이용할 수 있음.
- 프로그램은 서버에서 실행되고 **실행 결과(HTML, XML, JSON)**만 컴퓨터의 브라우저 또는 클라이언트 프로그램을 통해 보여짐.
- **데이터 입력, 메뉴선택, 버튼 클릭 등 사용자와의 상호작용 처리**를 위해 클라이언트에서 처리해야 되는 프로그램적인 요소도 있음. = javascript
- 웹 프로그램은 서버와 클라이언트의 조합에 의해 구현됨.

구분	종류
클라이언트 기술	HTML, 자바스크립트, CSS
서버 기술	서블릿, JSP, ASP.Net, PHP
클라이언트/서버 공통 기술	자바, C, C#.Net

c#

03. 스마트 시대의 웹 프로그래밍

■ 프레임워크(Framework)

■ 일반적인 개발의 문제점

- 프로그램의 규모 확대 -> 높은 생산성, 쉬운 유지 보수, 기능의 변경이 확장이 용이한 개발 기술 필요
- 개발방법론, 소프트웨어 디자인 패턴, 리팩토링, 프레임워크 등 소프트웨어 공학적 기술 등장

■ 프레임워크(Framework)는 무언가를 만들기 위한 구조 또는 틀.

- 소프트웨어적으로는 목적에 맞게 잘 설계된 구조와 미리 구현된 라이브러리가 포함된 소프트웨어 형태.
- 프레임워크를 사용하면 정해진 규격에 따라 프로그램 구조를 만들어야 하며,
- **개발자가 구현해야 하는 개발 방법론, 패턴, 아키텍처와 같은 처리는 프레임워크가 처리.**
- 여러 유틸리티 라이브러리도 제공하기 때문에 개발자는 적은 노력으로도 고품질의 소프트웨어 개발이 가능해짐.
- 대표적인 프레임워크는 **Spring Framework** - 웹 개발을 포함해 대규모 시스템 개발에 적합

웹 어플리케이션

■ 웹서버와 웹어플리케이션 서버

- 웹 서버와 WAS(Web Application Server)

- 웹 서버

정적인 콘텐츠(html, css, js)를 제공하는 서버 ex) Apache, Nginx

- WAS (Web Application Server)

☆ DB 조회나, 어떤 로직을 처리해야 하는 동적인 콘텐츠를 제공하는 서버
ex) Tomcat, Jeus -> java -> html

- 웹서버와 WAS의 차이

- 웹 서버와 WAS의 차이는 어떤 타입의 콘텐츠를 제공하느냐의 차이입니다.

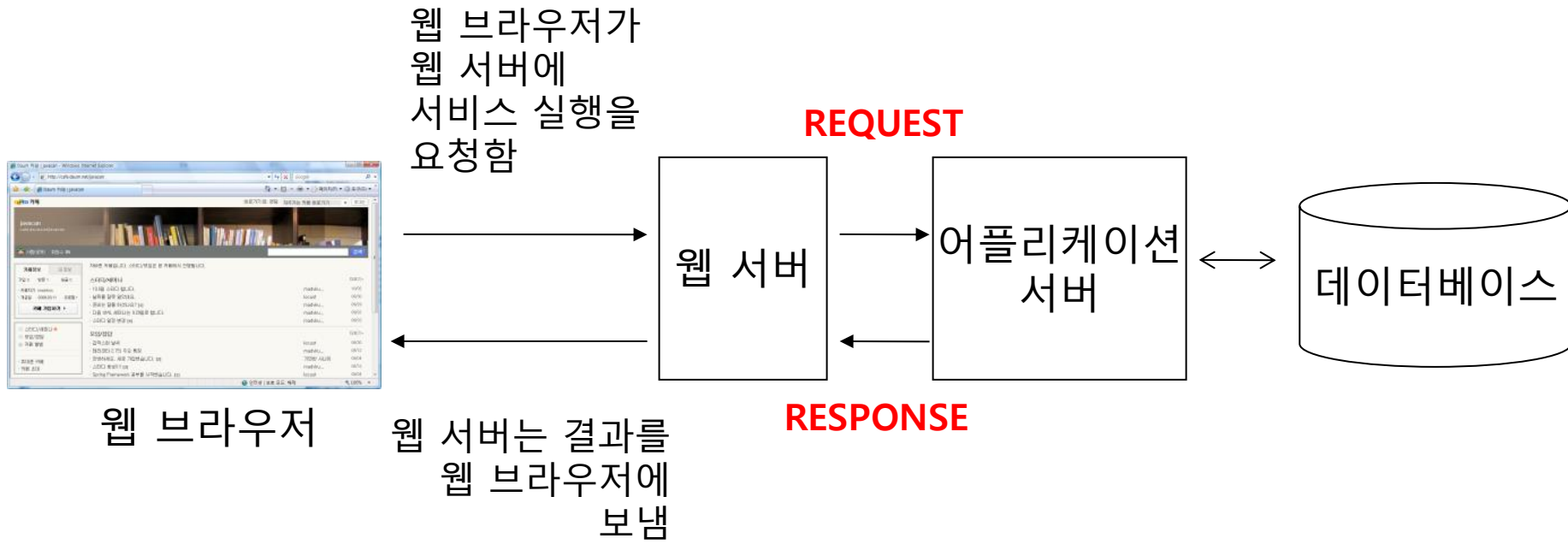
- 웹 서버와 WAS는 각각 독립적으로 구성할 수 있고, 대부분의 WAS는 정적인 콘텐츠를 제공해주고 있기 때문에, 웹 서버 없이 WAS만 존재할 수 있습니다.
tomcat

- 웹 서버 사용 이유

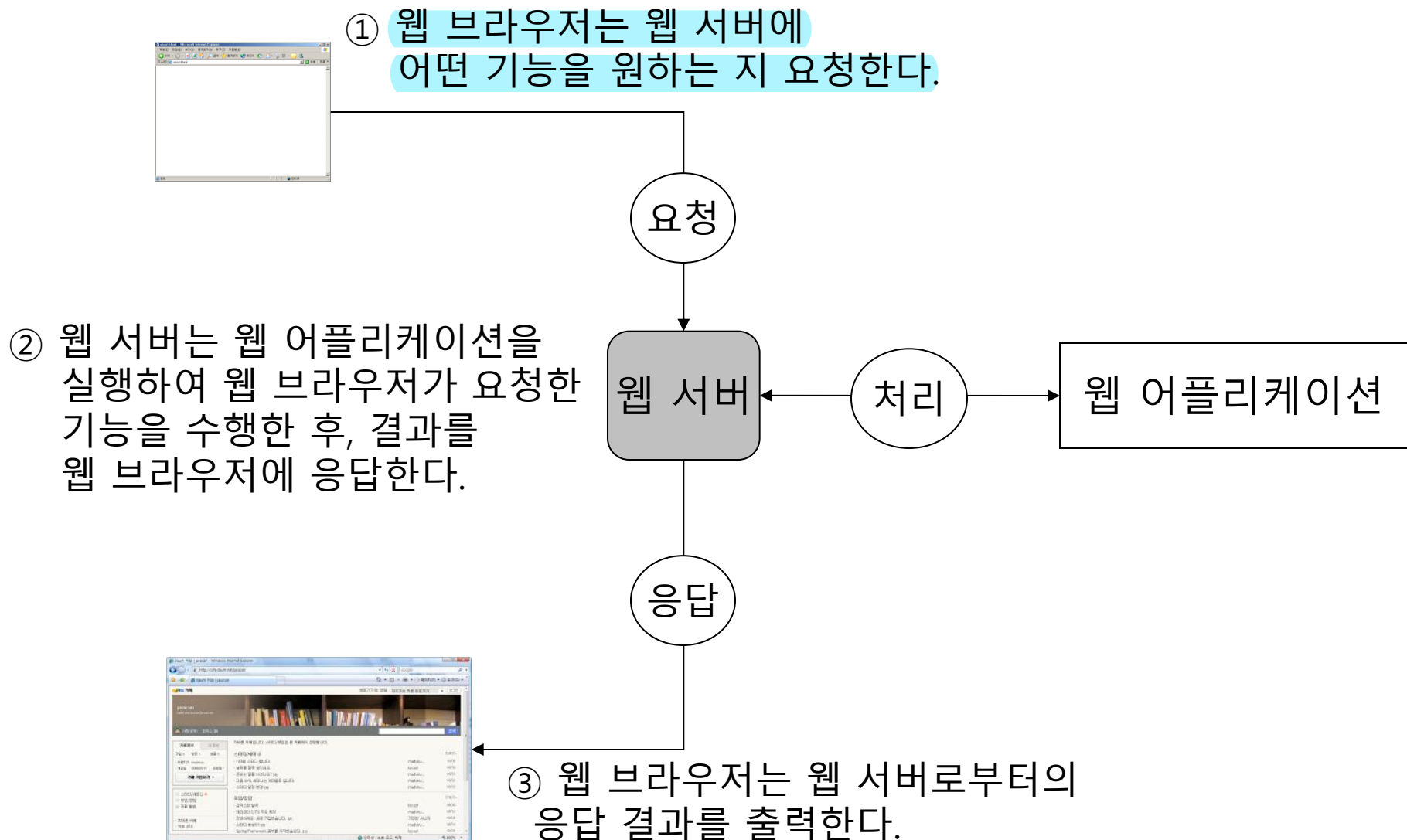
- WAS가 해야 할 일의 부담을 줄이기 위해서

- WAS의 환경설정 파일을 외부에 노출시키지 않도록 하기 위해서

■ 웹 어플리케이션의 구성 요소

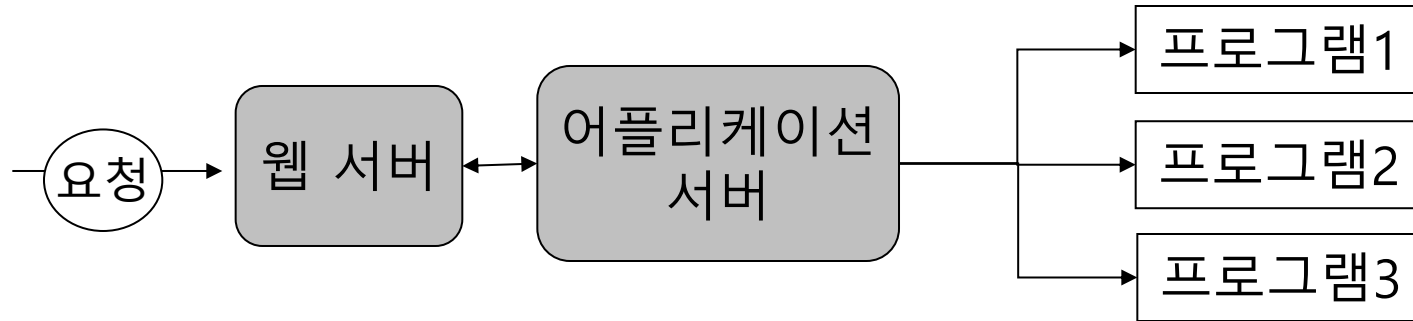


■ 웹 어플리케이션의 실행 순서



■ 어플리케이션 서버 방식

- 어플리케이션 서버 방식의 요청 처리



- 메모리 사용량 및 프로세스 관리 부하 감소로 전체적인 처리량 높음

■ 실행 코드 방식 vs 스크립트 방식

servlet
=>java

jsp
=>

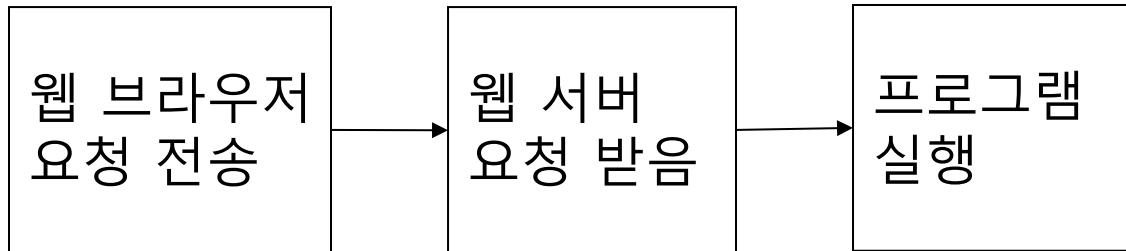
-

x

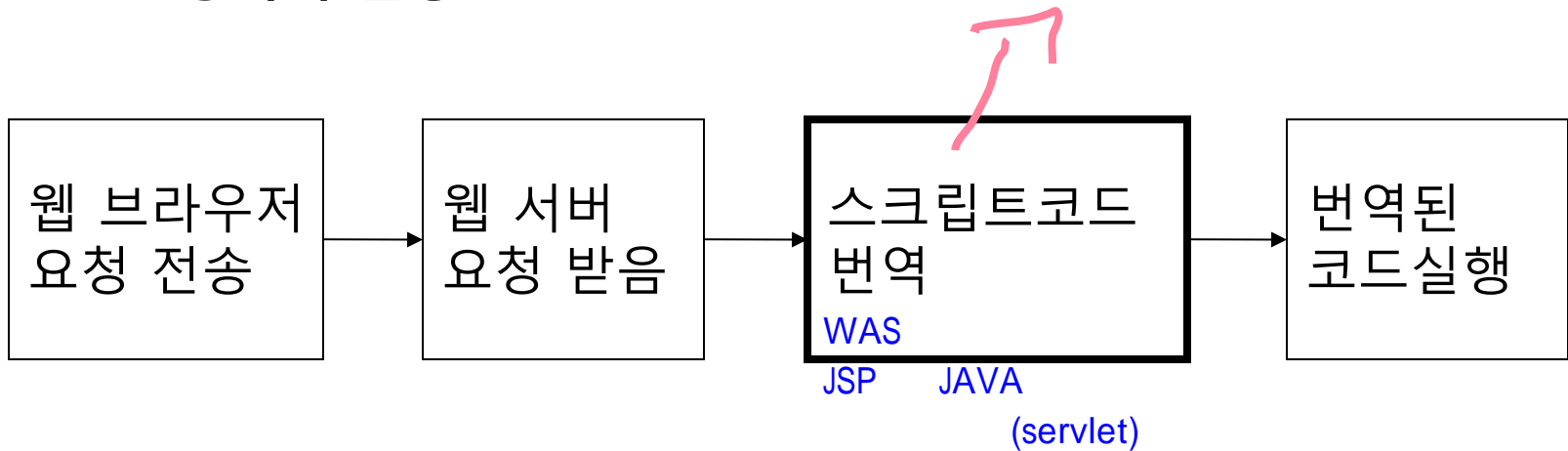
비교 항목	실행코드 방식	스크립트 방식
코드형태	컴파일 된 실행 프로그램	컴파일 되지 않은 스크립트 코드
실행방식	컴파일 된 기계어 코드 직접 실행	스크립트 코드를 해석한 뒤 실행
코드 변경	소스 코드를 다시 컴파일 해야 함.	스크립트 코드만 고치면 됨.
종류	C 기반 CGI 프로그램 java .	JSP, ASP.net, PHP, Ruby 등

■ 스크립트 방식과 실행 코드 방식의 실행 흐름

- 실행 코드 방식의 실행 흐름



- 스크립트 방식의 실행 흐름



■ URL

- Uniform Resource Locator
- 구성

[프로토콜]://[호스트][:포트][경로][파일명][.확장자][쿼리문자열]

- 예: `http://www.google.com/search?hl=en&q=jsp&aq=f&oq=`
 - 프로토콜: http
 - 호스트: www.google.com
 - 포트: 80 (http 프로토콜의 기본 포트)
 - 경로: /search
 - 쿼리문자열: hl=en&q=jsp&aq=f&oq=
- URL은 웹 어플리케이션에 요청을 구분하기 위한 용도로 사용됨

■ 서블릿과 JSP

- 자바를 만든 Sun에서 정한 웹 개발 표준

- 서블릿(Servlet) : 실행 코드 방식의 특징
- JSP(JavaServer Pages) : 스크립트 코드 방식의 특징

mvc -> spring

- JSP의 특징

jsp =

- 자바 기반 스크립트 언어
 - 자바의 기능을 그대로 사용 가능
- HTTP에 대한 클라이언트의 요청 처리/응답
- 웹 어플리케이션에서 결과 화면을 생성할 때 주로 사용

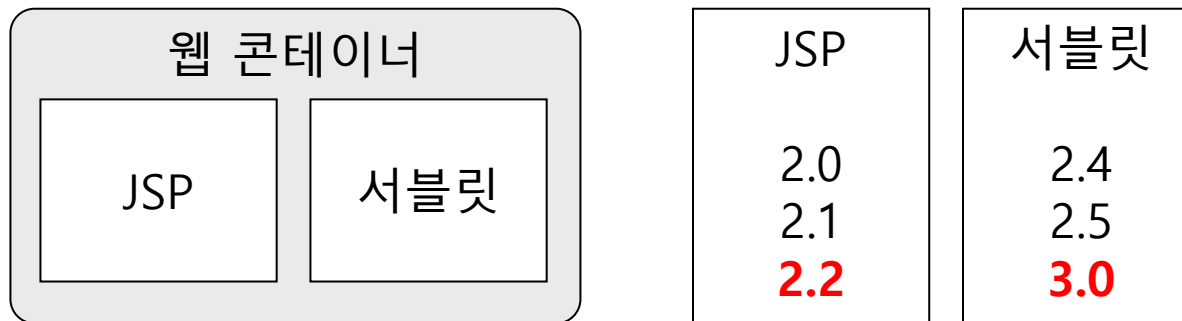
jsp
mvc -> spring



HTML, CSS, javascript

■ 웹 컨테이너

- 웹 어플리케이션을 실행할 수 있는 컨테이너
- JSP와 서블릿을 실행해 줌



- 주요 웹 컨테이너

– 톰캣(Tomcat) : <http://tomcat.apache.org/>

+

웹 어플리케이션 개발 환경 설정

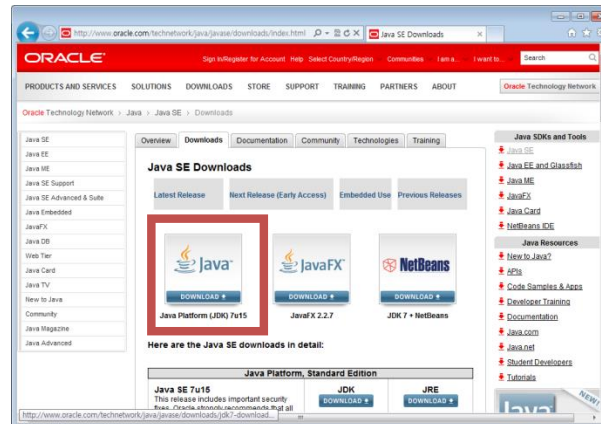
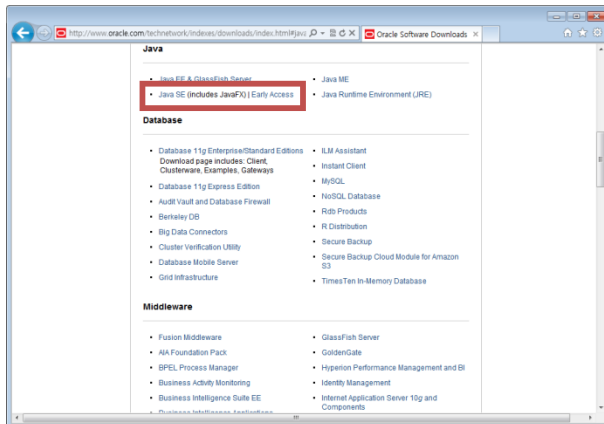
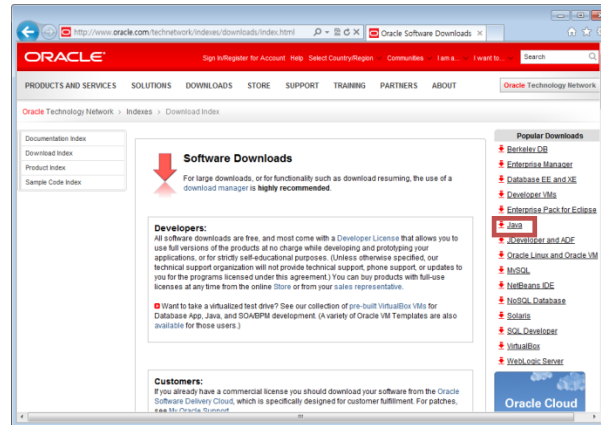
01. JSP 개발환경 개요

■ 개발환경

항목	필요 프로그램
자바 개발환경	JDK
JSP 운영환경(서블릿 컨테이너)	아파치 톰캣
통합 개발환경	이클립스, intellij

02. JSP 개발환경 구축

❶ <http://www.oracle.com/>에 접속하여 상단 메뉴에서 [DOWNLOADS]를 클릭한다.
Downloads 페이지에서 [Java]를 클릭한 후 다시 [Java SE]를 클릭하고 [Java Platform (JDK)]를 클릭한다.



02. JSP 개발환경 구축

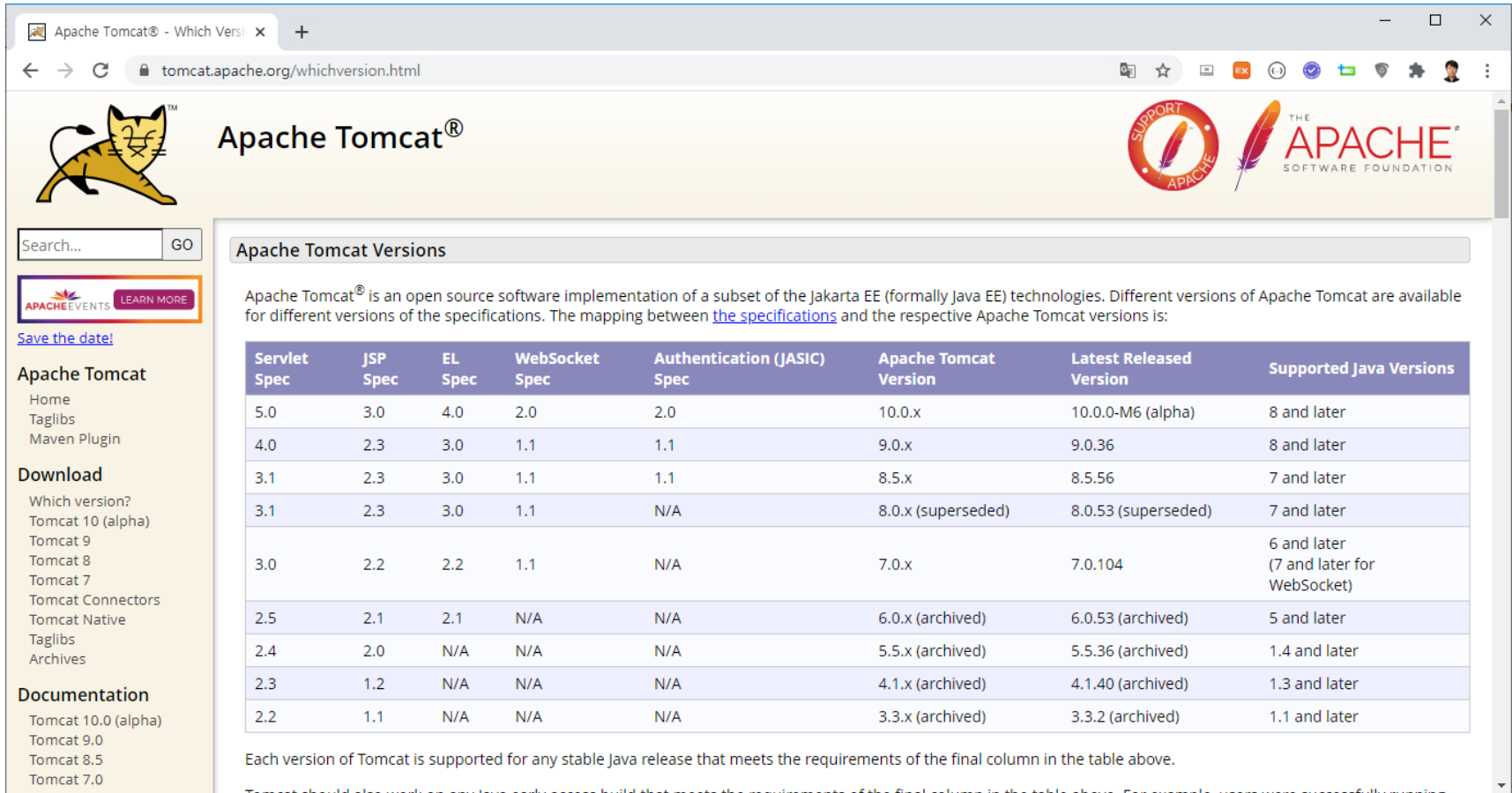
2. JSP 운영환경 구축: 아파치 TOMCAT

- 아파치 톰캣(Apache Tomcat)은 서블릿 컨테이너라고도 한다.
- 서블릿 컨테이너는 서블릿과 JSP를 실행할 수 있는 환경을 말한다.
기본적으로 웹 서버(apache)의 기능도 포함하고 있으며 자바로 개발되어 있다.
- 개발용이 아닌 실제 서비스용으로 웹 서버를 운영하려면
→ 아파치 웹 서버를 설치하고 서블릿 컨테이너로 TOMCAT 을 사용할 수 있도록 설정.
- JSP를 학습하려는 것이 목적이므로 아파치 웹 서버를 별도로 설치하지 않고 TOMCAT 이 제공하는 웹 서버 기능을 그대로 사용함.

02. JSP 개발환경 구축

■ 아파치 톰캣 설치하기

<http://tomcat.apache.org/>에 접속한 후 [Download] 메뉴에서 [Tomcat 8.5]을 클릭한다.
그리고 파일 중 [64-bit Windows zip]를 클릭하여 다운로드 한다.



The screenshot shows the Apache Tomcat website's version selection page. The page features the Tomcat logo, a search bar, and a sidebar with navigation links. The main content area is titled 'Apache Tomcat Versions' and contains a table mapping various specifications to Tomcat versions. Below the table, there is a note about Java version requirements.

Apache Tomcat Versions

Apache Tomcat® is an open source software implementation of a subset of the Jakarta EE (formerly Java EE) technologies. Different versions of Apache Tomcat are available for different versions of the specifications. The mapping between [the specifications](#) and the respective Apache Tomcat versions is:

Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Authentication (JASIC) Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
5.0	3.0	4.0	2.0	2.0	10.0.x	10.0.0-M6 (alpha)	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.36	8 and later
3.1	2.3	3.0	1.1	1.1	8.5.x	8.5.56	7 and later
3.1	2.3	3.0	1.1	N/A	8.0.x (superseded)	8.0.53 (superseded)	7 and later
3.0	2.2	2.2	1.1	N/A	7.0.x	7.0.104	6 and later (7 and later for WebSocket)
2.5	2.1	2.1	N/A	N/A	6.0.x (archived)	6.0.53 (archived)	5 and later
2.4	2.0	N/A	N/A	N/A	5.5.x (archived)	5.5.36 (archived)	1.4 and later
2.3	1.2	N/A	N/A	N/A	4.1.x (archived)	4.1.40 (archived)	1.3 and later
2.2	1.1	N/A	N/A	N/A	3.3.x (archived)	3.3.2 (archived)	1.1 and later

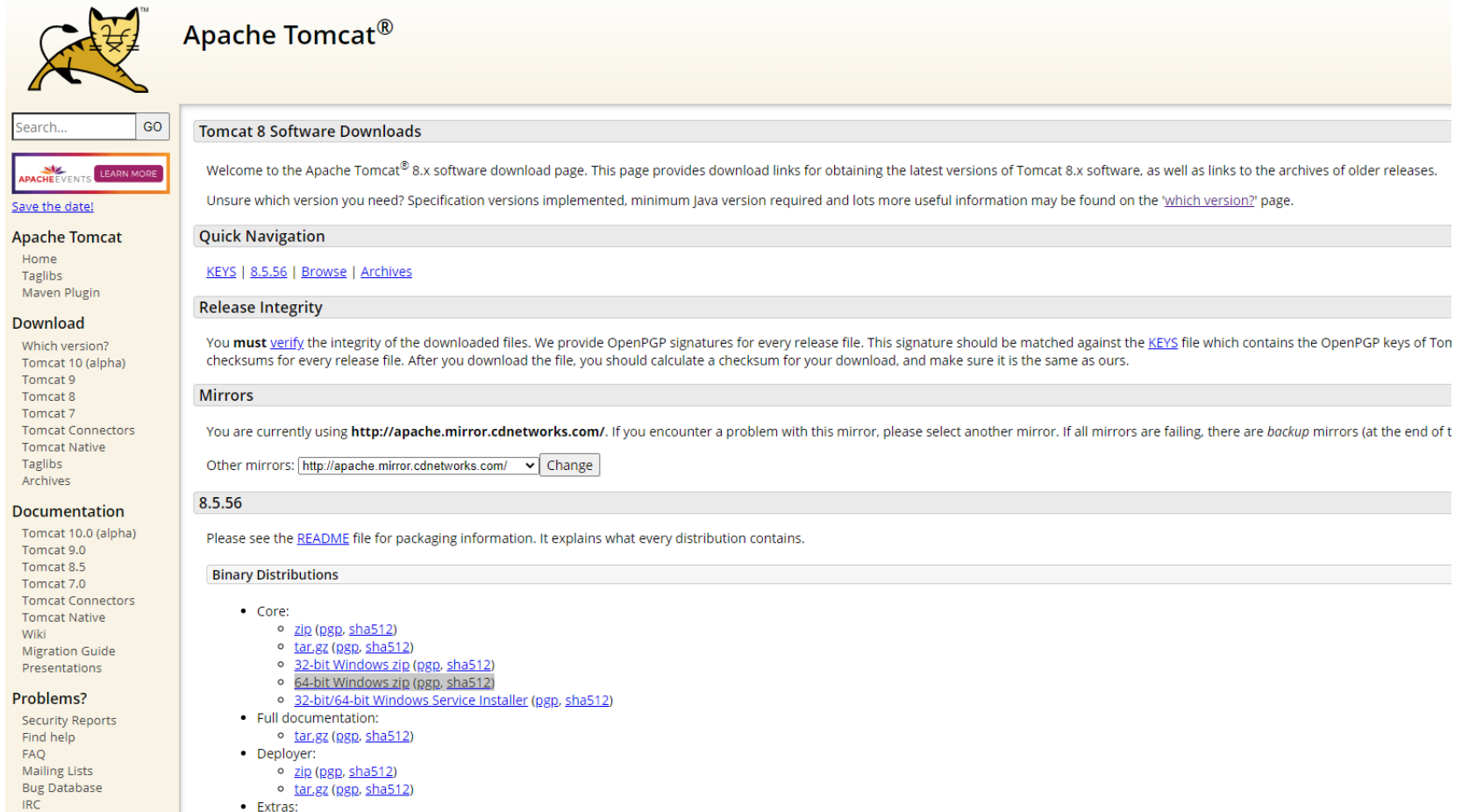
Each version of Tomcat is supported for any stable Java release that meets the requirements of the final column in the table above.

Tomcat should also work on any Java early access build that meets the requirements of the final column in the table above. For example, users were successfully running

02. JSP 개발환경 구축

■ 아파치 톰캣 설치하기

<http://tomcat.apache.org/>에 접속한 후 [Download] 메뉴에서 [Tomcat 8.5]을 클릭한다.
그리고 파일 중 [64-bit Windows zip]를 클릭하여 다운로드 한다.



Apache Tomcat®

Search... GO

[APACHE EVENTS](#) [LEARN MORE](#)

[Save the date!](#)

Apache Tomcat

- Home
- Taglibs
- Maven Plugin

Download

- Which version?
- Tomcat 10 (alpha)
- Tomcat 9
- Tomcat 8
- Tomcat 7
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

Documentation

- Tomcat 10.0 (alpha)
- Tomcat 9.0
- Tomcat 8.5
- Tomcat 7.0
- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide
- Presentations

Problems?

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

Tomcat 8 Software Downloads

Welcome to the Apache Tomcat® 8.x software download page. This page provides download links for obtaining the latest versions of Tomcat 8.x software, as well as links to the archives of older releases.

Unsure which version you need? Specification versions implemented, minimum Java version required and lots more useful information may be found on the ["which version?"](#) page.

Quick Navigation

[KEYS](#) | [8.5.56](#) | [Browse](#) | [Archives](#)

Release Integrity

You **must** [verify](#) the integrity of the downloaded files. We provide OpenPGP signatures for every release file. This signature should be matched against the [KEYS](#) file which contains the OpenPGP keys of Tomcat releases for every release file. After you download the file, you should calculate a checksum for your download, and make sure it is the same as ours.

Mirrors

You are currently using <http://apache.mirror.cdnetworks.com/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the list).

Other mirrors:

8.5.56

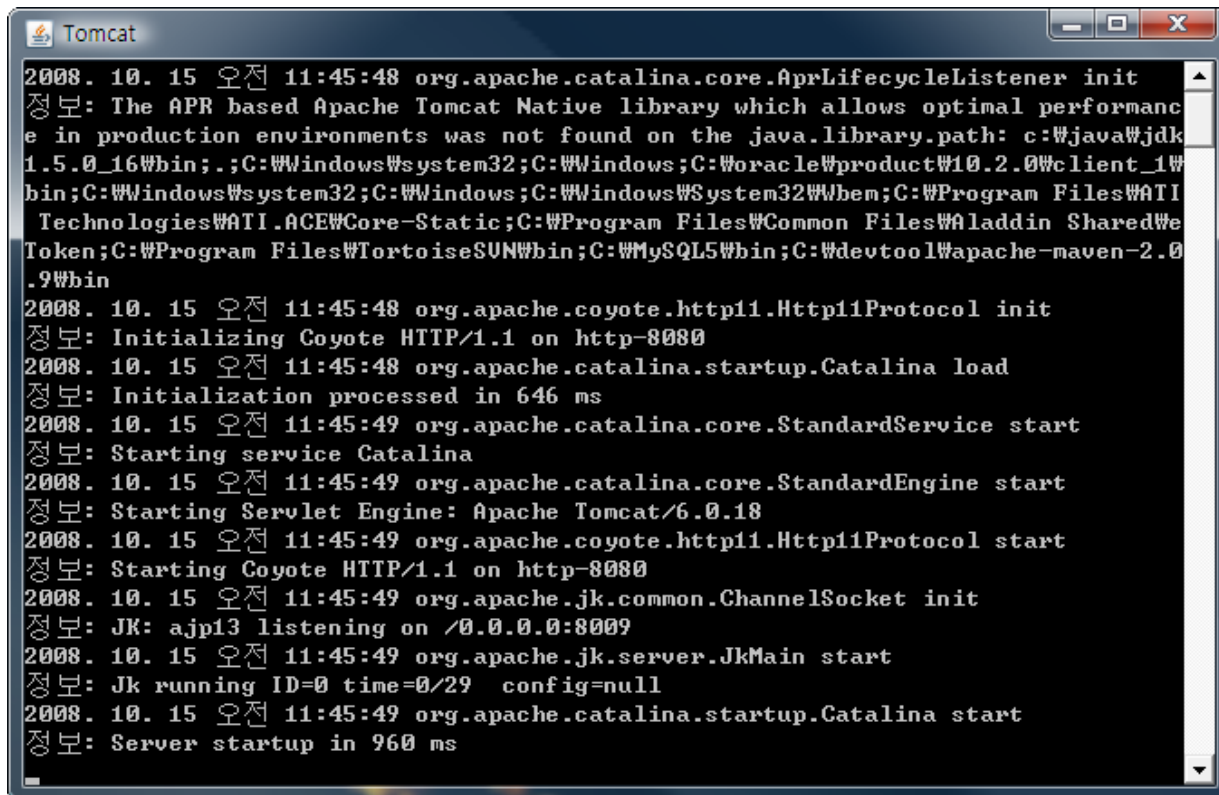
Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
 - [32-bit Windows zip \(pgp, sha512\)](#)
 - [64-bit Windows zip \(pgp, sha512\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
 - [tar.gz \(pgp, sha512\)](#)
- Deployer:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
- Extras:

■ 톰캣 실행

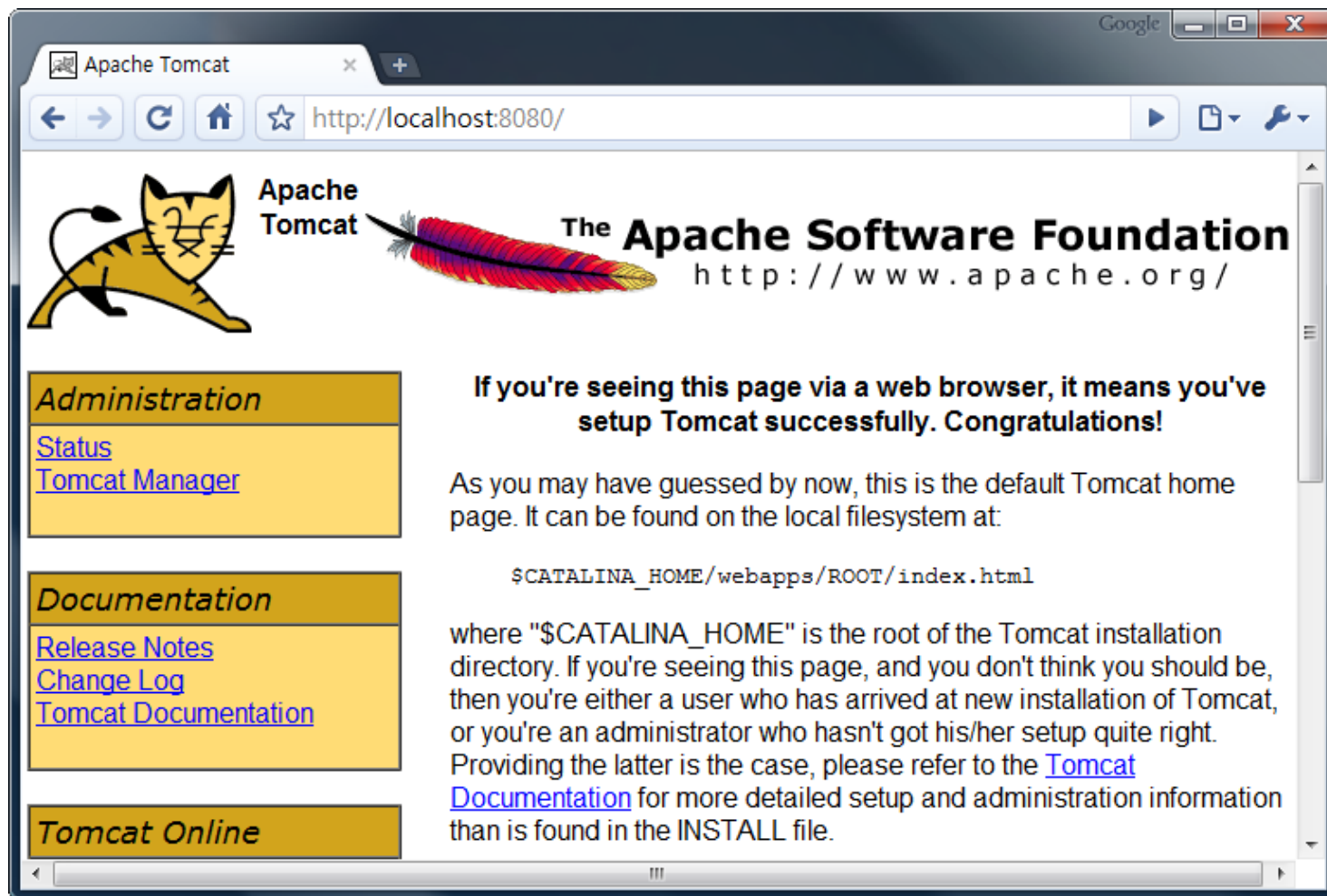
- 톰캣 설치 디렉터리/bin 에 위치한 실행 스크립트
 - startup.bat – 톰캣을 독립 프로세스로 시작
 - shutdown.bat – 실행된 톰캣을 종료시킨다.
 - catalina.bat – 톰캣을 시작하거나 종료한다.



```
Tomcat
2008. 10. 15 오전 11:45:48 org.apache.catalina.core.AprLifecycleListener init
정보: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: c:\java\jdk1.5.0_16\bin;.;C:\Windows\system32;C:\Windows;C:\Oracle\product\10.2.0\client_1\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files\ATI Technologies\ATI.ACE\Core-Static;C:\Program Files\Common Files\Aladdin Shared\Token;C:\Program Files\TortoiseSUN\bin;C:\MySQL5\bin;C:\devtool\apache-maven-2.0.9\bin
2008. 10. 15 오전 11:45:48 org.apache.coyote.http11.Http11Protocol init
정보: Initializing Coyote HTTP/1.1 on http-8080
2008. 10. 15 오전 11:45:48 org.apache.catalina.startup.Catalina load
정보: Initialization processed in 646 ms
2008. 10. 15 오전 11:45:49 org.apache.catalina.core.StandardService start
정보: Starting service Catalina
2008. 10. 15 오전 11:45:49 org.apache.catalina.core.StandardEngine start
정보: Starting Servlet Engine: Apache Tomcat/6.0.18
2008. 10. 15 오전 11:45:49 org.apache.coyote.http11.Http11Protocol start
정보: Starting Coyote HTTP/1.1 on http-8080
2008. 10. 15 오전 11:45:49 org.apache.jk.common.ChannelSocket init
정보: JK: ajp13 listening on /0.0.0.0:8009
2008. 10. 15 오전 11:45:49 org.apache.jk.server.JkMain start
정보: Jk running ID=0 time=0/29 config=null
2008. 10. 15 오전 11:45:49 org.apache.catalina.startup.Catalina start
정보: Server startup in 960 ms
```

■ 웹 브라우저 실행 화면

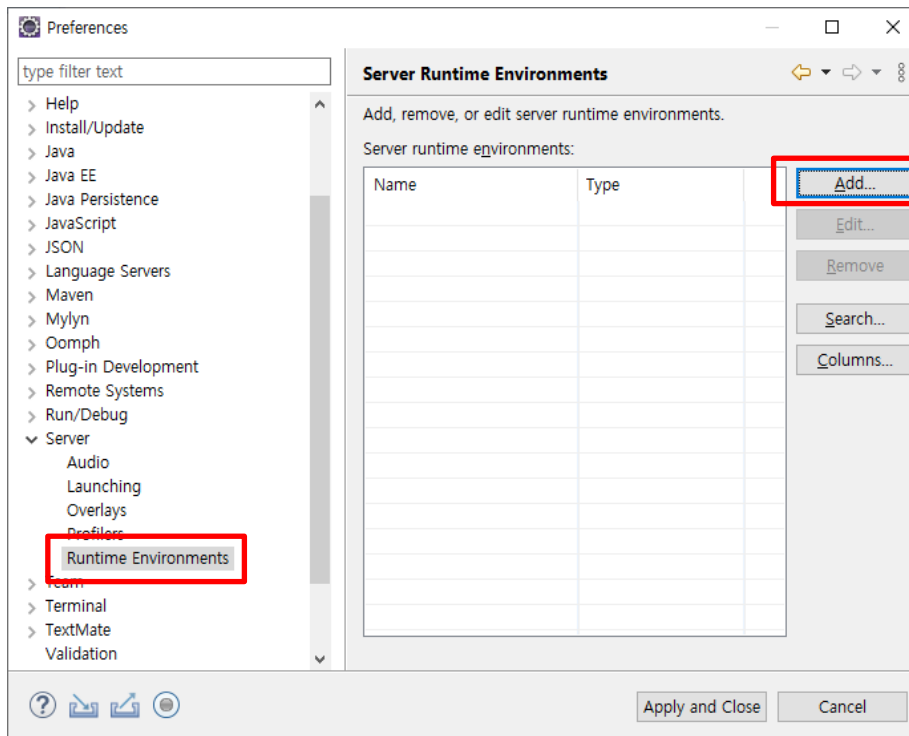
- <http://localhost:8080>



03. 이클립스 기본 환경 설정

■ Server Runtime Environment 설정

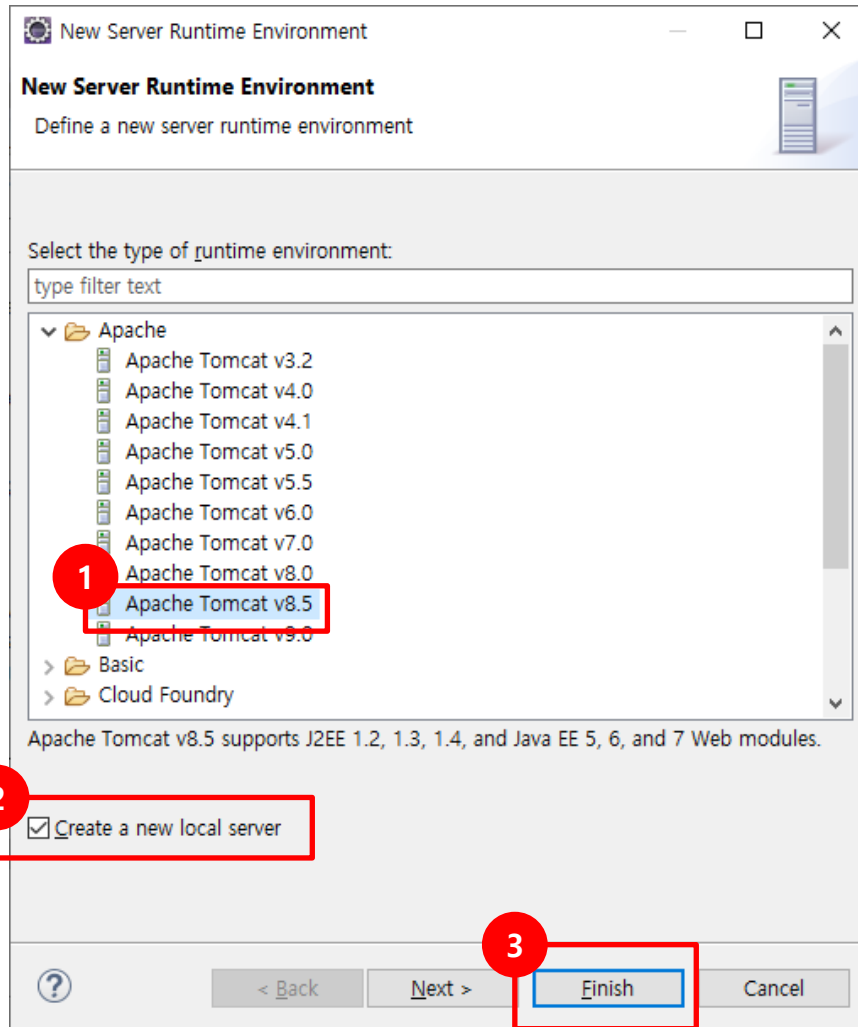
- [Serrver] → [Runtime Environment]
- WAS 설정을 위해서는 WAS가 설치되어 있어야 합니다.
- 설정 과정에서도 설치가 가능합니다.



03. 이클립스 기본 환경 설정

■ Server Runtime Environment 설정

- [Serrver] → [Runtime Environment] → [Add]

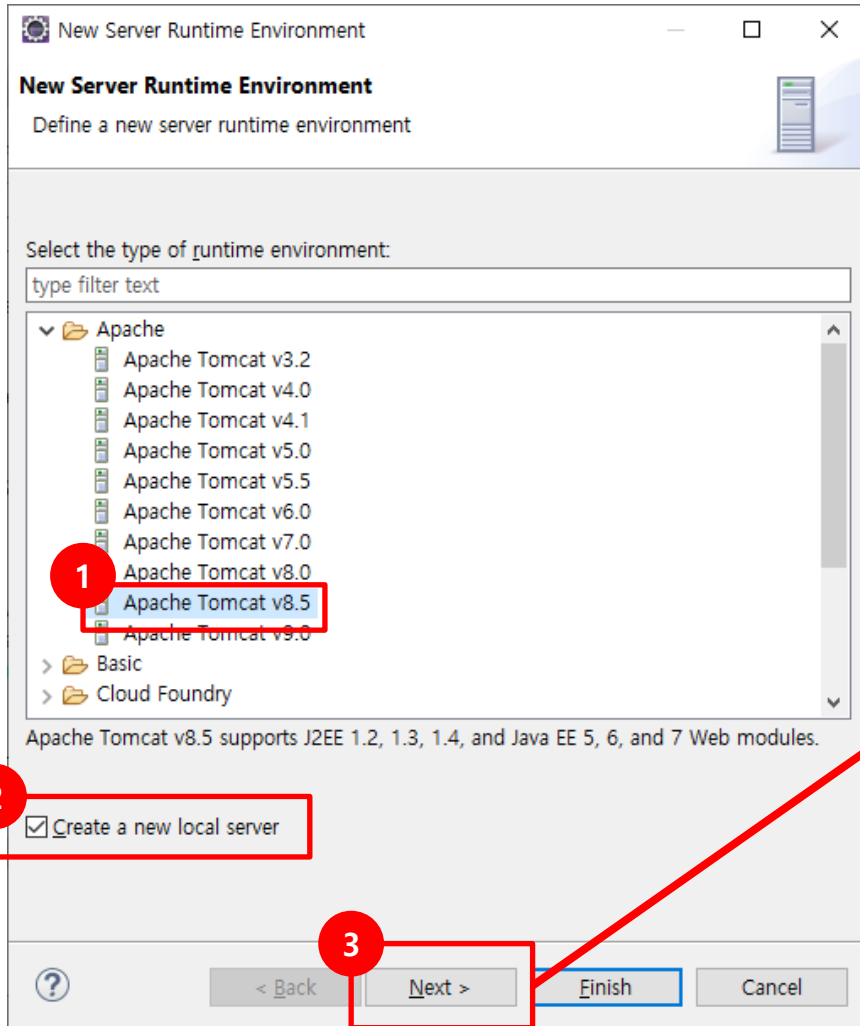


1. 설치한 AWS 의 버전을 확인하고 선택해줍니다.
2. 설치된 WAS에서 Local에서 서버를 생성합니다.
3. 완료 버튼

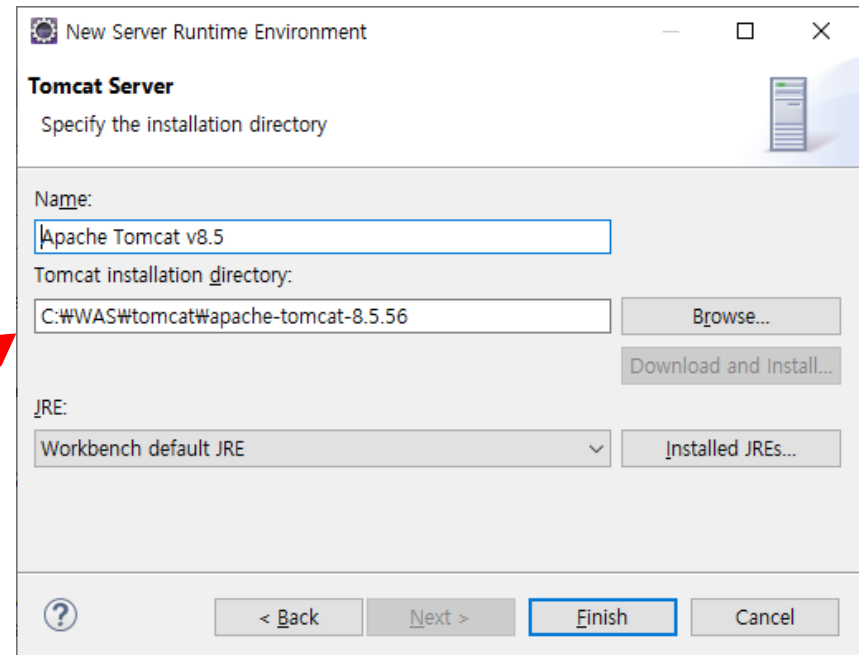
03. 이클립스 기본 환경 설정

■ Server Runtime Environment 설정

■ [Serrver] → [Runtime Environment] → [Add]

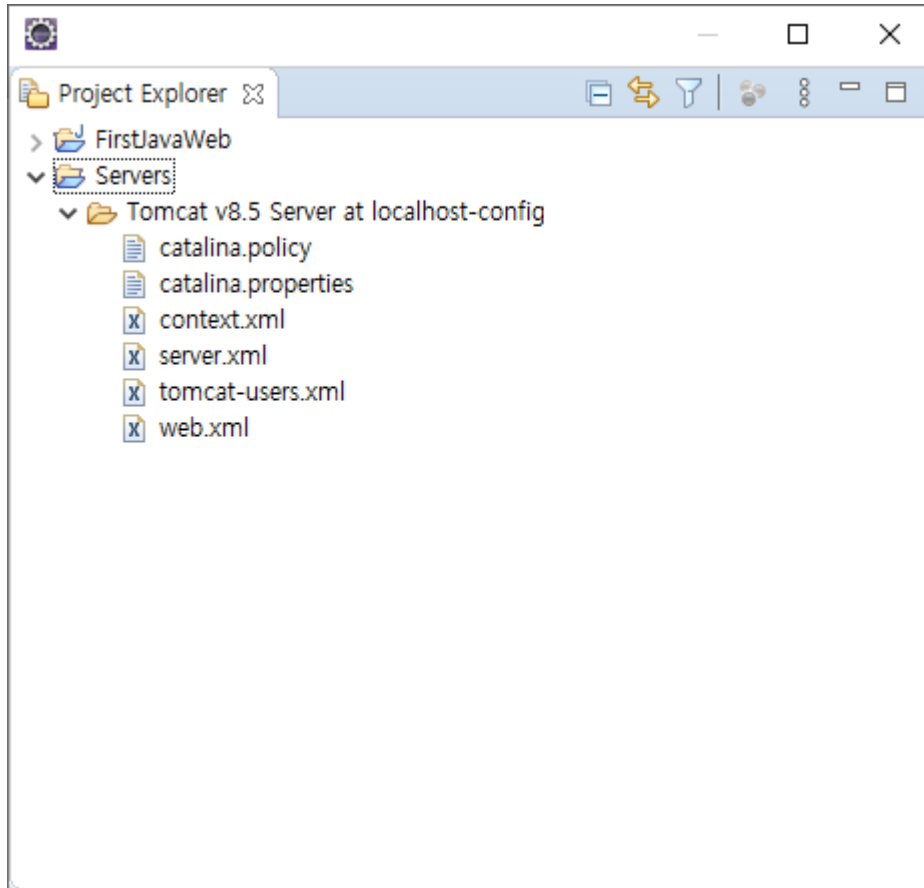


1. 설치한 AWS 의 버전을 확인하고 선택해줍니다.
2. 설치된 WAS에서 Local에서 서버를 생성합니다.
3. 다음 버튼 클릭
4. localServer의 이름과 Tomcat이 설치된 경로를 등록해줍니다.



03. 이클립스 기본 환경 설정

■ Server Runtime Environment 설정



03. 이클립스 기본 환경 설정

■ 자바 소스 기본 주석 설정하기

- 프로그램에서 주석은 유지보수에 매우 중요한 역할을 함. 자바의 경우 Javadoc 주석을 이용해 클래스 API 문서를 손쉽게 만들고 관리할 수 있다.
- 이클립스에서 소스 생성시 자동으로 만들어주는 주석의 내용을 설정을 통해 조정 가능.
- [Java] → [Code Style] → [Code Template]을 선택하고 [Comments] 트리를 확장해 세부 설정.
- 파일 주석 설정
 - [Comments] → [Files] → <Edit> 버튼
 - 새로 생성한 모든 자바 소스 파일 맨 상단에 생성되는 주석
 - 보통 파일 이름, 작성일, 파일 설명을 기입
 - <Insert Variable> 버튼을 눌러 \${} 형태의 변수 값 입력이 가능함.

03. 이클립스 기본 환경 설정

```
/**
```

```
 * @FileName : ${file_name}
 * @Project : ${project_name}
 * @Date : ${date}
 * @작성자 : ${user}
 * @변경이력 :
 * @프로그램 설명 :
 */
```

```
/**
```

```
 * @Method Name : ${enclosing_method}
 * @작성일 : ${date}
 * @작성자 : ${user}
 * @변경이력 :
 * @Method 설명 :
 * ${tags}
 */
```

```
/**
```

```
 ${date} : Current date (현재 날짜)
 ${dollar} : The dollar symbol (달러문양)
 ${enclosing_type} : The type enclosing the method (선택된 메소드의 타입)
 ${file_name} : Name of the enclosing compilation (선택된 편집파일 이름)
 ${package_name} : Name of the enclosing package (선택된 패키지 이름)
 ${project_name} : Name of the enclosing project (선택된 프로젝트 이름)
 ${tags} : Generated Javadoc tags (@param, @return...) (Javadoc 태그 생성)
 ${time} : Current time (현재 시간)
 ${todo} : Todo task tag ('해야할일'태그 생성)
 ${type_name} : Name of the current type (현재 타입의 이름)
 ${user} : User name (사용자 이름)
 ${year} : Current year (현재 연도)
 */
```

03. 이클립스 기본 환경 설정

■ 컴파일러 버전 설정하기

- [Java] → [Compiler]
- 호환성 유지를 위해서는 이클립스에서 기본 JDK 버전 설정이 필요함.
- [Compiler compliance level] 항목에서 원하는 버전으로 수정함.

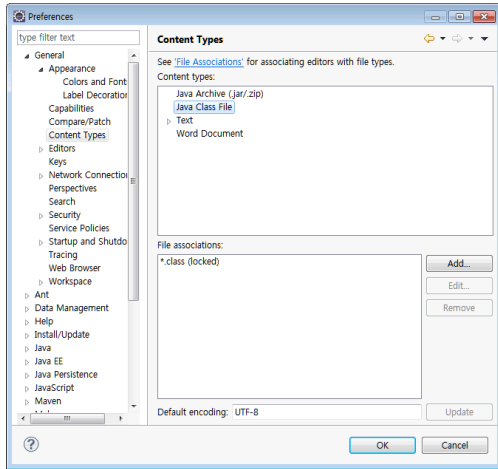
■ 텍스트 인코딩 설정하기

- 프로그램 개발 시 한글 사용은 여러 문제를 발생시키는 원인이 됨.
- 다국어를 지원하는 프로그램 개발을 위해서는 유니코드 기반인 UTF-8 의 사용이 권장됨.
- 안드로이드 등 모바일 개발에도 유니코드가 기본임.

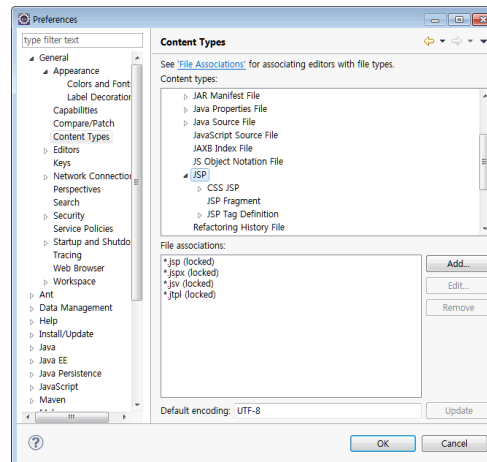
❶ [General] → [Workspace] → Text file encoding 항목을 Other 로 변경한 뒤 UTF-8로 설정

03. 이클립스 기본 환경 설정

② 자바 클래스 인코딩 설정 : [General] → [Content Types] → Java Class File에 대한 Default encoding 값을 UTF-8 로 입력 후 <Update> 버튼

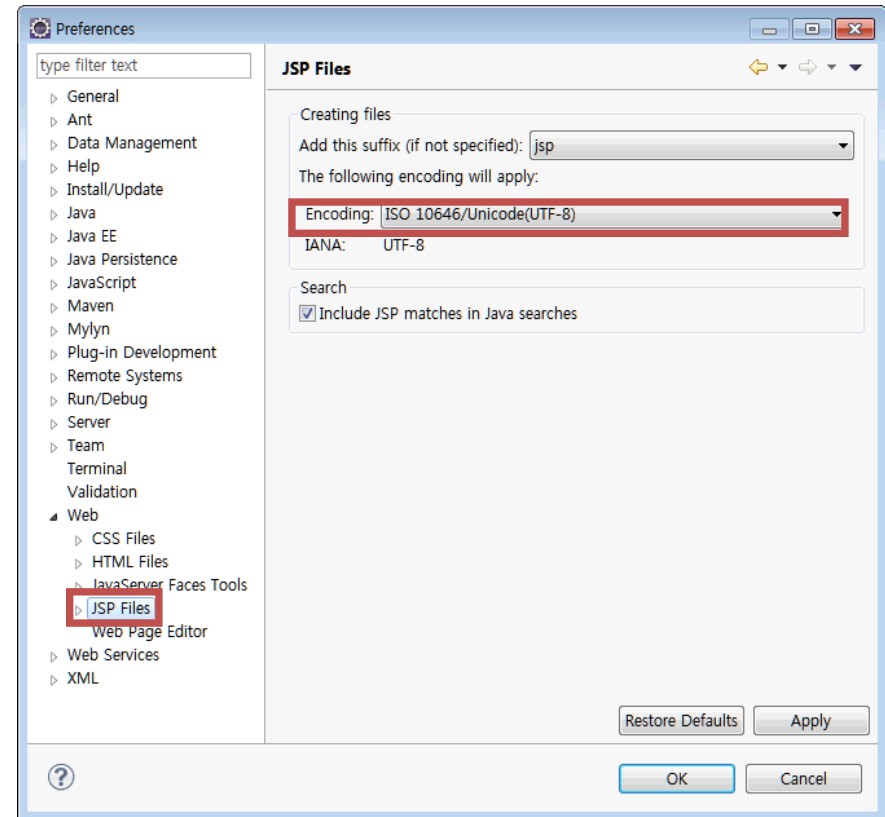
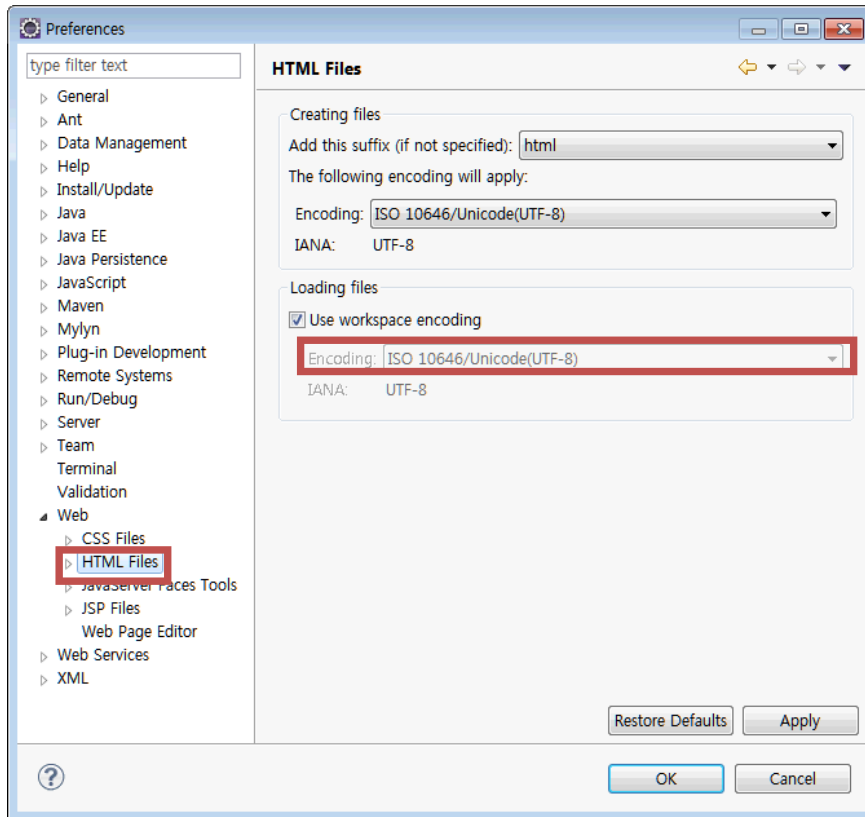


③ JSP 파일 인코딩 설정 : [General]→[Content Types] → [Text] → JSP 항목을 UTF-8 로 변경 후 <Update> 버튼



03. 이클립스 기본 환경 설정

④ CSS / HTML / JSP 코드 인코딩 설정 : [Web] → [HTML Files], [JSP Files] 항목을 ISO 10646/Unicode(UTF-8)로 변경



■ 웹 어플리케이션 개발 퀵 스타트

- 1단계, 간단한 JSP 작성하기
- 2단계, 간단한 서블릿 작성
- 3단계, 실행

■ 1단계, JSP 작성

- testWnow.jsp 파일 작성

```
<%@page import="java.util.Date"%>
```

디렉티브

```
<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>
```

```
<%
```

```
    Date now = new Date();
```

스크립트릿(scriptlet)

```
%>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>현재 시간</title>
```

```
</head>
```

```
<body>
```

현재 시각:

```
<%=now%>
```

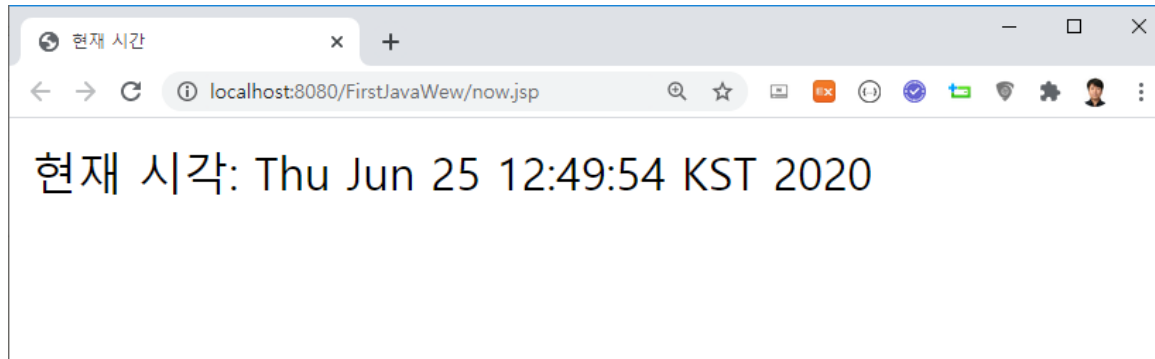
표현식(expression)

```
</body>
```

```
</html>
```

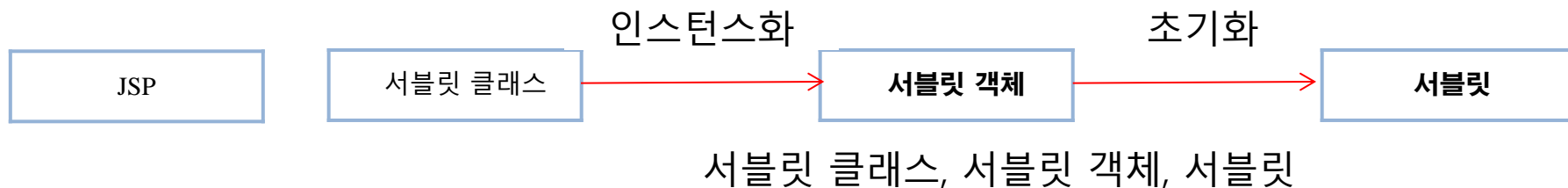
■ 2단계, JSP 실행

- JSP 실행
 - `http://localhost:8080/now.jsp`
[툼캣]WwebappsWFirstJavaWebWnow.jsp가 실행됨



■ 2단계, 간단한 서블릿 작성

- 서블릿이란 서블릿 클래스를 상속해서 만들어진 객체이다. -> html
- 웹 컨테이너는 서블릿 클래스를 가지고 서블릿 객체를 만든 다음 그 객체를 초기화해서 웹 서비스를 할 수 있는 상태로 만드는데, 이 작업을 거친 서블릿 객체만 서블릿이라고 할 수 있다.

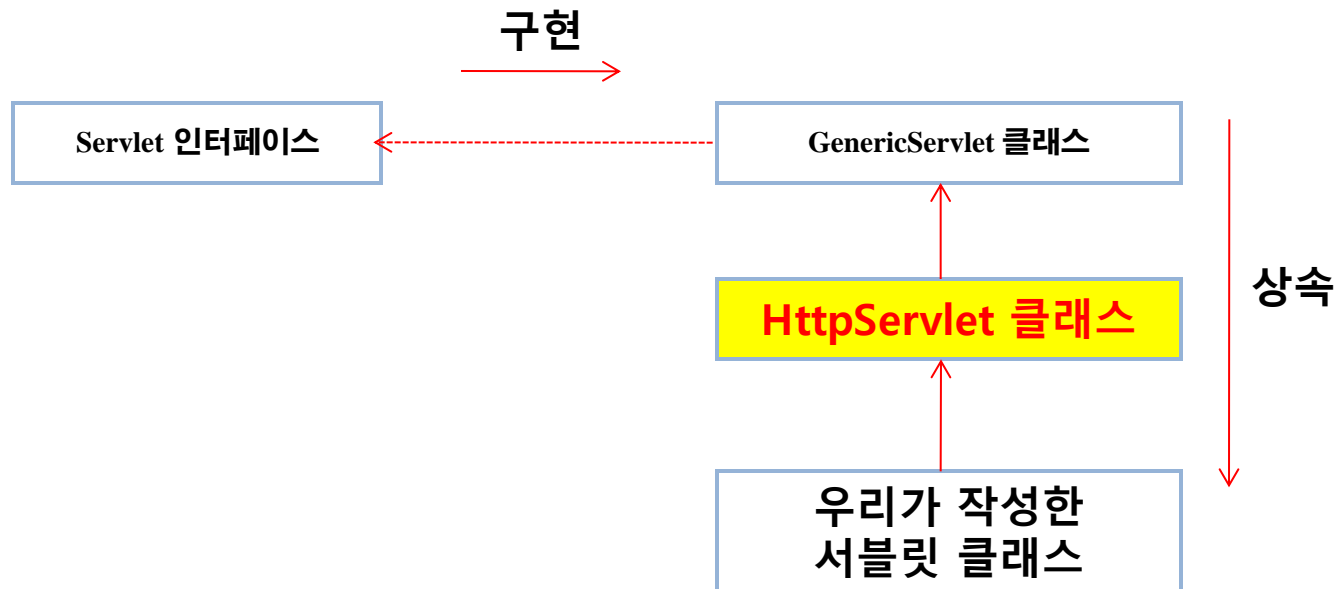


-> 가 가

()

■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스를 작성할 때 지켜야 할 규칙 세 가지
 - 서블릿 클래스는 **javax.servlet.http.HttpServlet** 클래스를 상속해서 만듬.
 - doGet 또는 doPost 메서드 안에 웹 브라우저로부터 요청이 왔을 때 해야 할 작업 코드를 기술해야 한다
 - HTML 문서는 doGet, doPost 메서드의 두 번째 매개변수인 response 객체를 이용해서 응답 결과를 출력해야 한다



■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스 작성하기

- 서블릿 클래스를 작성할 때 지켜야 할 첫 번째 규칙:

`javax.servlet.http.HttpServlet` 클래스를 상속받도록 만들어야 한다.

그리고 `public` 클래스로 만들어야 한다.

- 서블릿 클래스 안에 `doGet` 또는 `doPost` 메서드를 선언해야 하며, 이 두 메서드는

`javax.servlet.http.HttpServletRequest`와 `javax.servlet.http.HttpServletResponse` 타입의 파라미터를 받고, 메서드 밖으로 `javax.servlet.ServletException`과 `java.io.IOException`을 던질 수 있도록 선언해야 한다.

- 이클립스와 같은 도구를 이용하면 간단하게 클래스를 만들 수 있음.

■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스 작성하기

- doGet 메서드를 작성할 때는 다음과 같은 골격을 만드는 것으로 시작한다.

```
public class HundredServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
}
```

HttpServlet 클래스의 doGet 메서드와
리턴 타입, 파라미터 변수, 익셉션 타입이 동일해야 한다.

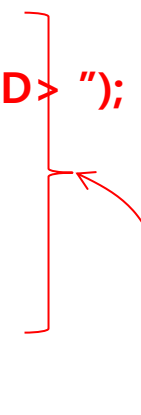
- doGet 메서드를 public으로 선언해야 하는 이유는 웹 컨테이너가 웹 브라우저로부터 요청을 받아서 메서드를 호출할 때 필요하기 때문이다.
- doGet 메서드의 throws 절에 있는 ServletException과 IOException이 필요치 않으면 생략할 수도 있다. 하지만 다른 익셉션을 추가할 수는 없다.
이유는 오버라이딩 규칙을 따라야 하기 때문입니다.

■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스 작성하기

- 계산 결과를 웹 브라우저로 출력하는 코드

```
public class HundredServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        int total = 0;  
        for (int cnt = 1; cnt < 101; cnt++)  
            total += cnt;  
        PrintWriter out = response.getWriter();  
        out.println( "<HTML> ");  
        out.println( "<HEAD><TITLE>Hundred Servlet</TITLE> </HEAD> ");  
        out.println( "<BODY> ");  
        out.printf( "1 + 2 + 3 + ... + 100 = %d ", total);  
        out.println( "</BODY> ");  
        out.println( "</HTML> ");  
    }  
}
```



계산 결과를 HTML로 만
들어서 웹 브라우저로 출
력하는 명령문

■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스 등록하기

- 서블릿 클래스는 JSP 페이지와 달리 설치뿐만 아니라 등록 과정도 필요하다.
- 웹 애플리케이션의 디플로이먼트 디스크립터 파일에 등록해야 한다.
- 웹 애플리케이션의 디플로이먼트 디스크립터 파일란 웹 애플리케이션 디렉터리의 WEB-INF 서브디렉터리 아래 있는 web.xml이라는 이름의 파일을 말한다.
예: 톰캣의 webapps\examples\ WEB-INF 디렉터리에 있는 web.xml 파일
- XML 파일이고, 텍스트 에디터를 이용해서 열어 볼 수 있다.

■ 2단계, 간단한 서블릿 작성

- 서블릿 클래스 등록하기
 - 서블릿 3.0버전부터는 `@WebServlet` 이노테이션을 사용하면, 웹 컨테이너가 자동 등록.

■ 서블릿 클래스의 작성, 컴파일, 설치, 등록

```
package test;

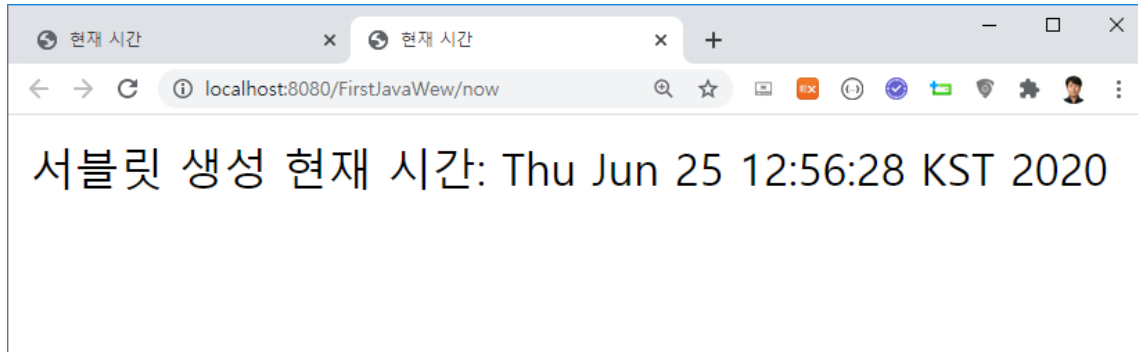
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

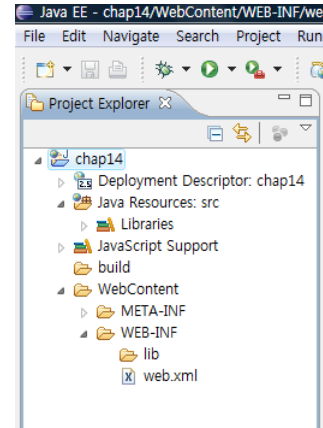
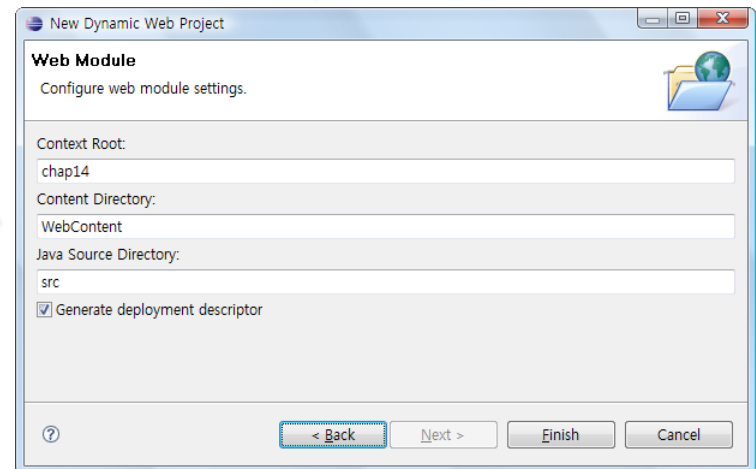
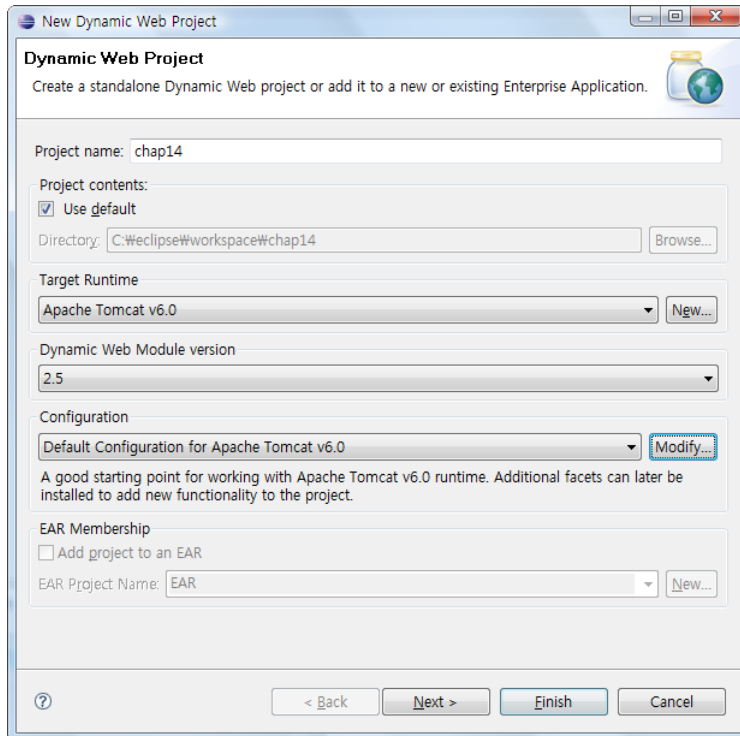
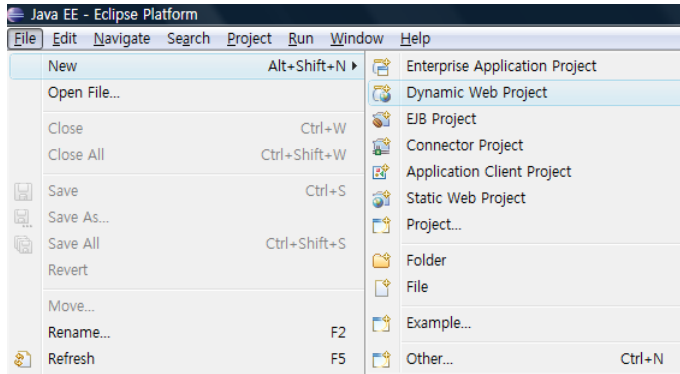
@WebServlet("/now")
public class NowServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=euc-kr");
        Date now = new Date();
        PrintWriter writer = response.getWriter();
        writer.println("<html>");
        writer.println("<head><title>현재 시간</title></head>");
        writer.println("<body>");
        writer.println("서블릿 생성 현재 시간:");
        writer.println(now.toString());
        writer.println("</body>");
        writer.println("</html>");
        writer.close();
    }
}
```

■ 3. 실행

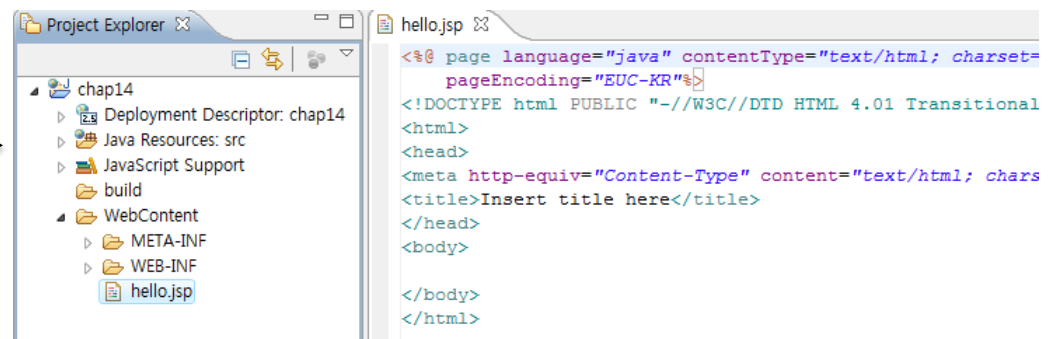
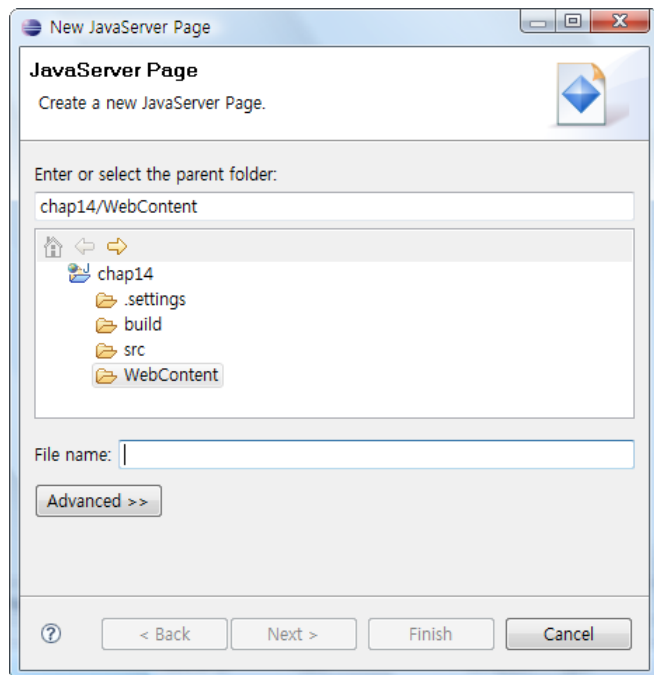
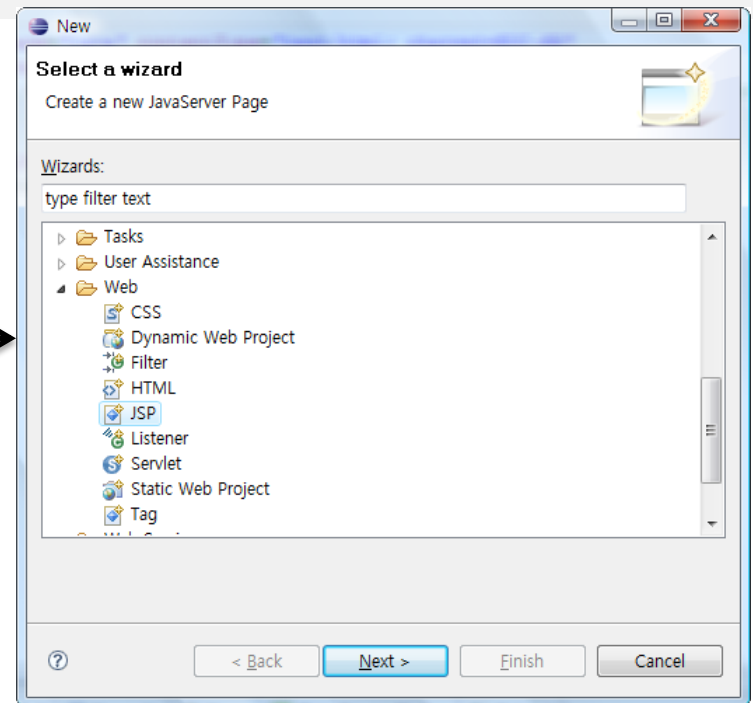
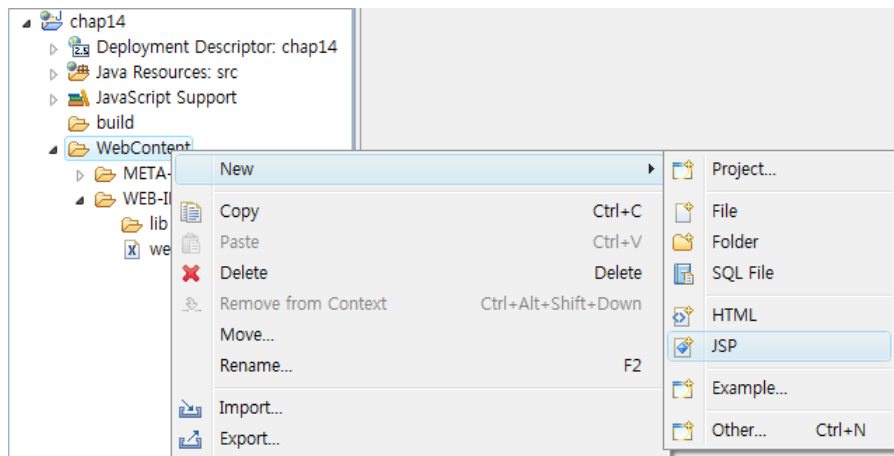
- <http://localhost:8080/now>



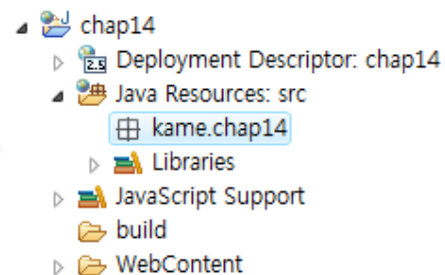
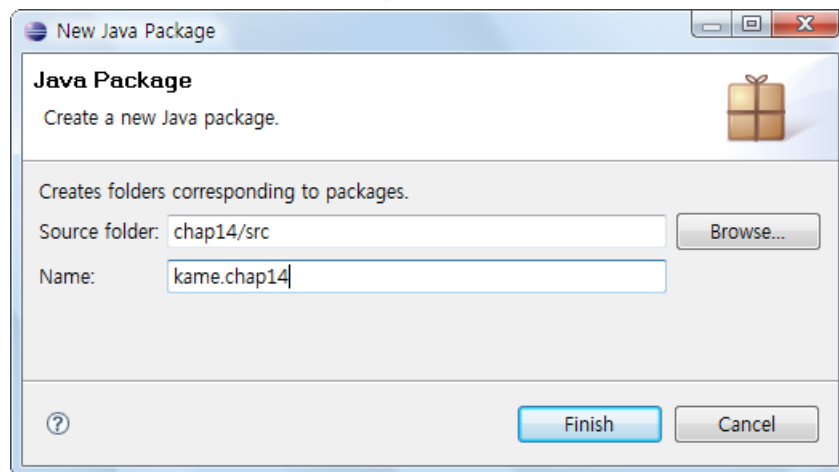
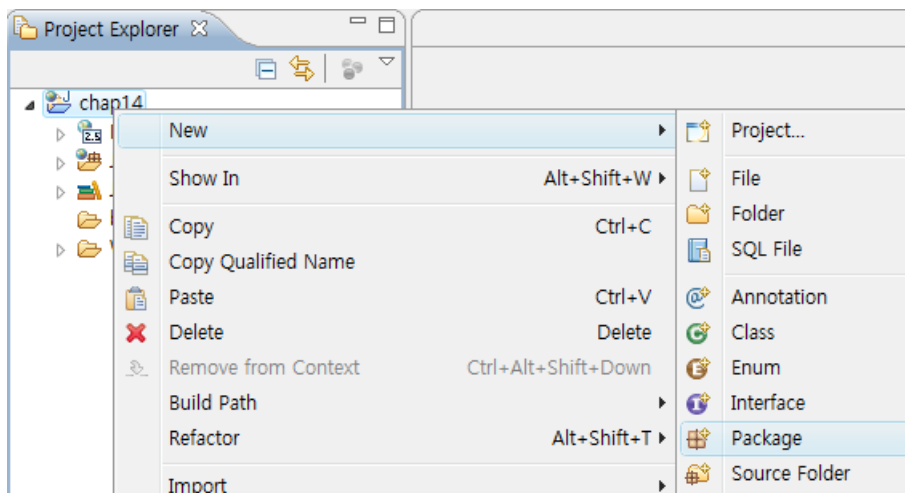
웹 프로젝트 생성



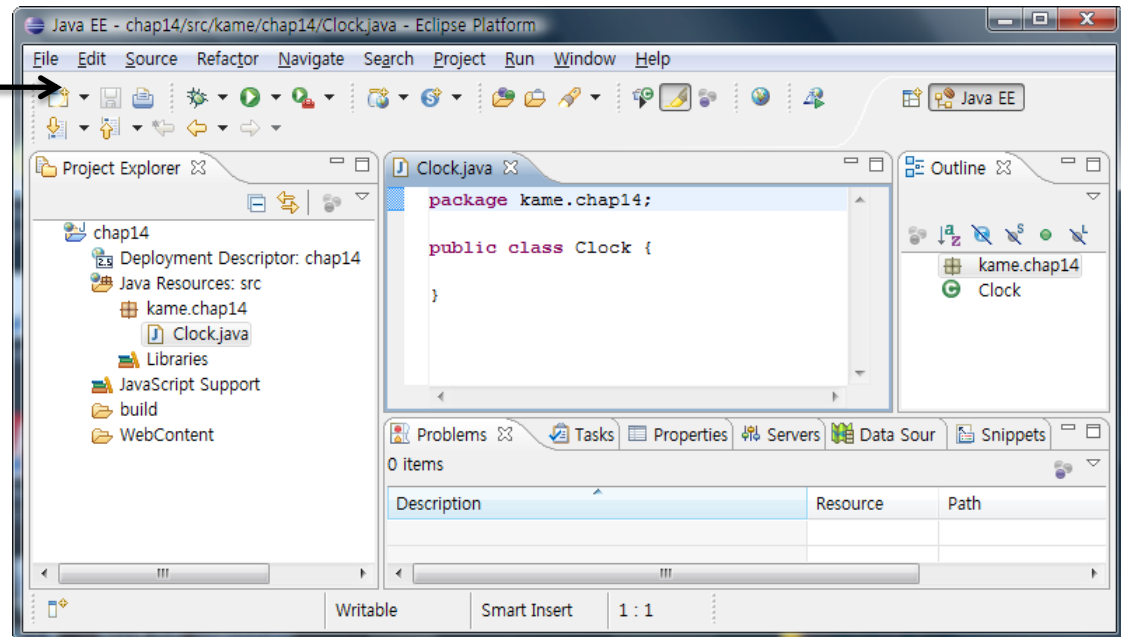
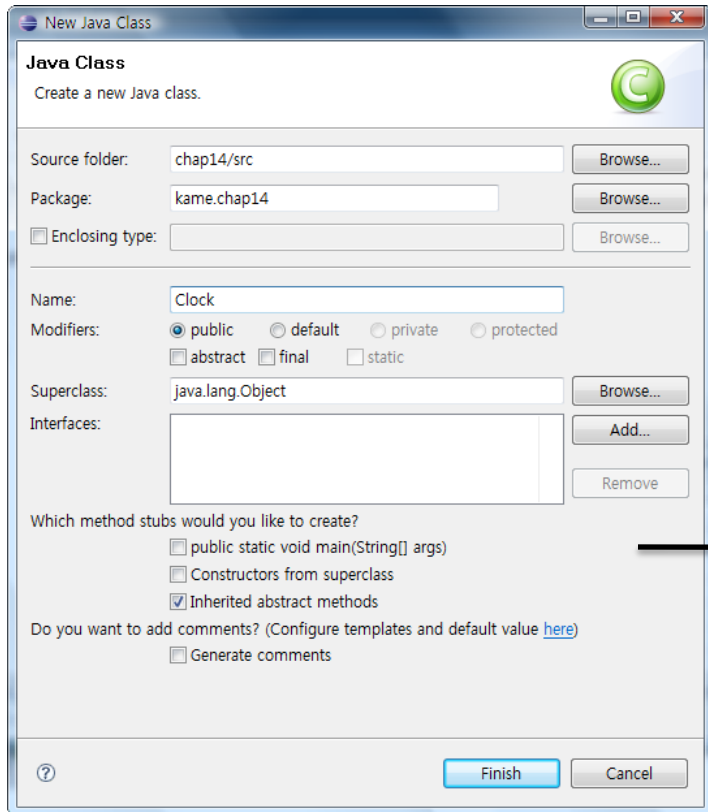
JSP 추가



■ 패키지 생성 추가

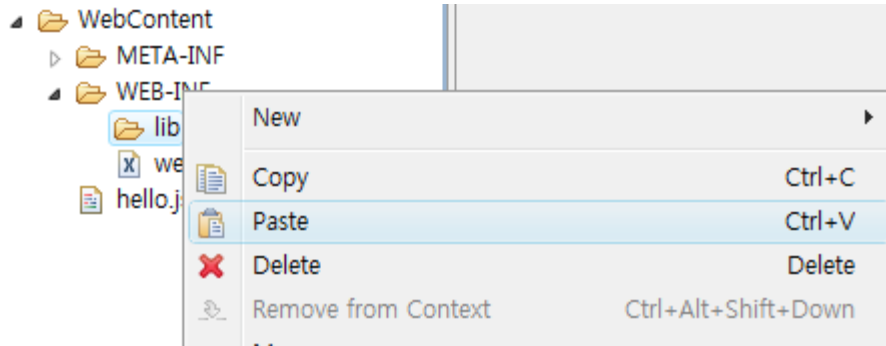


자바 클래스 추가

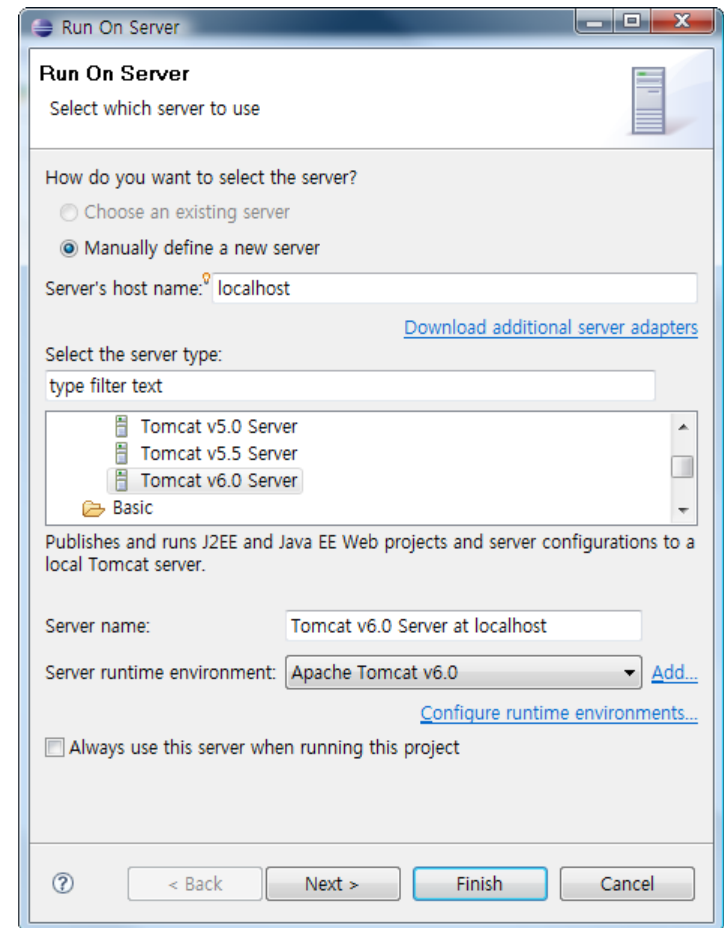
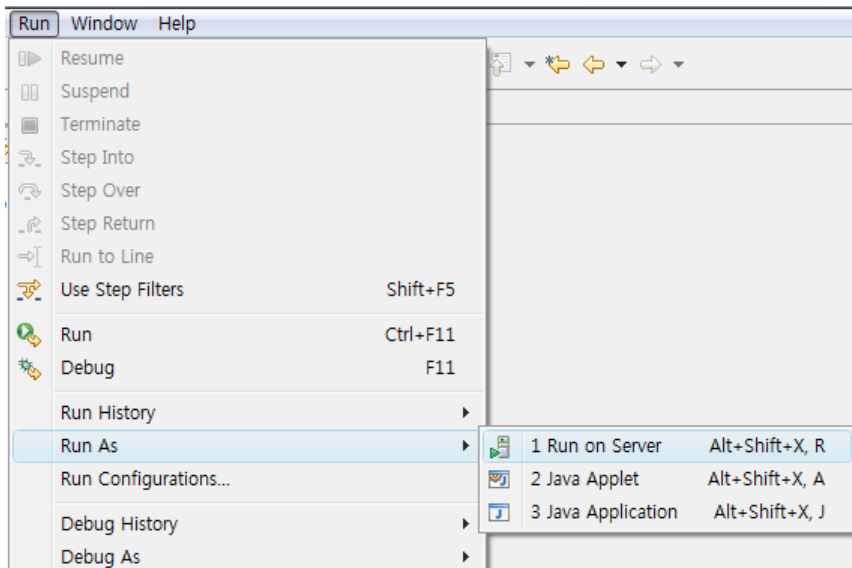


■ jar 파일 추가

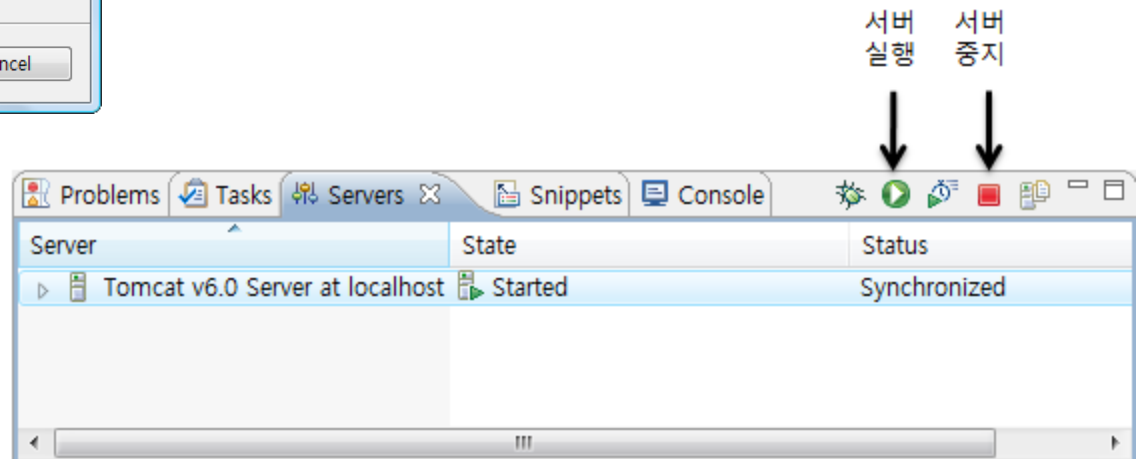
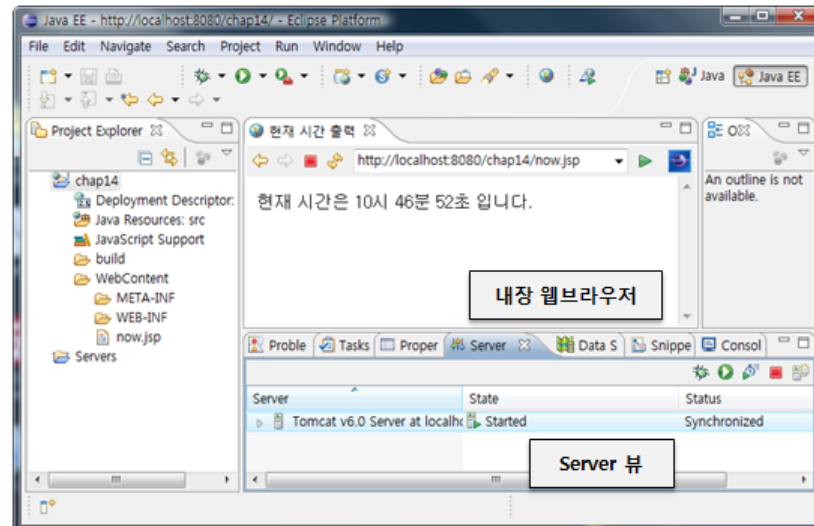
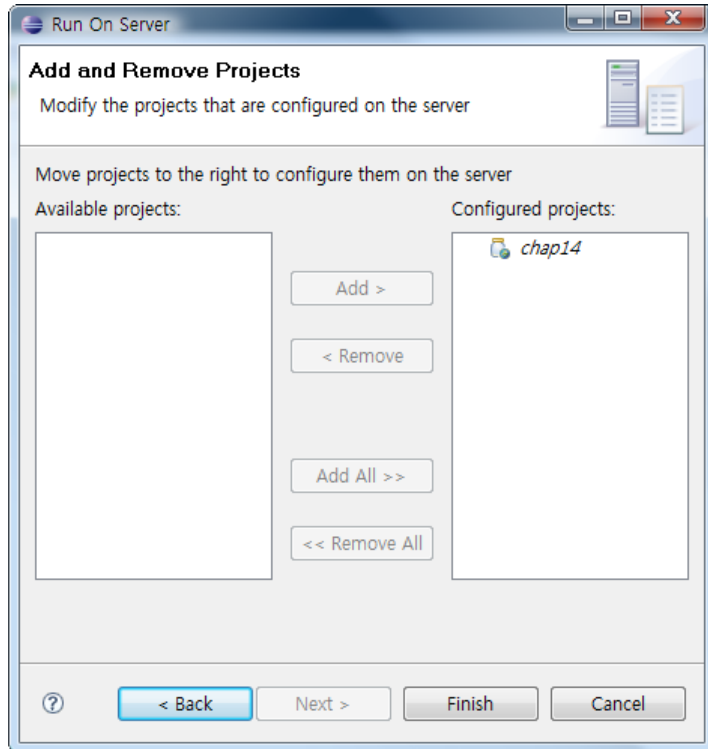
- WEB-INF/lib에 jar 파일 복사



■ 웹 어플리케이션 실행 1

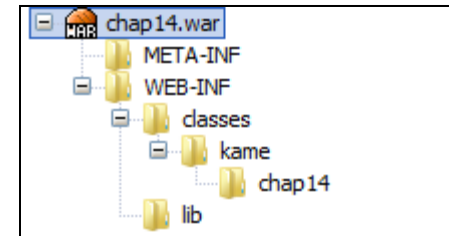
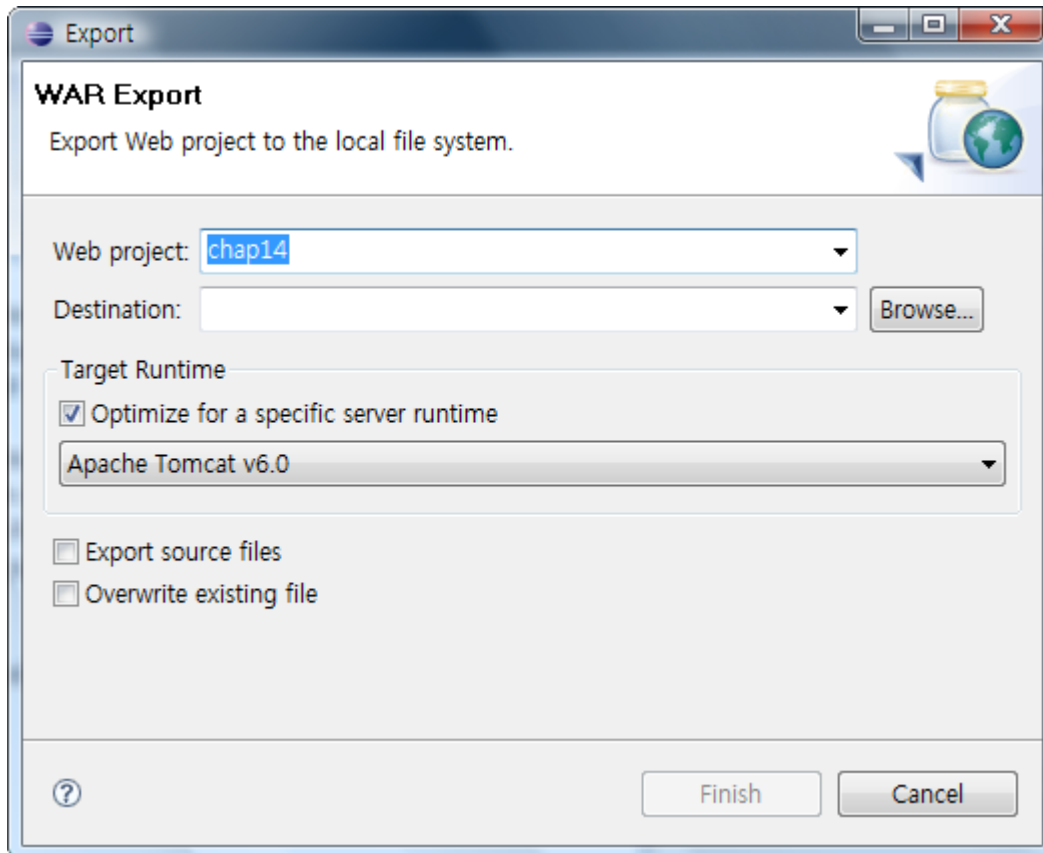


■ 웹 어플리케이션 실행 2



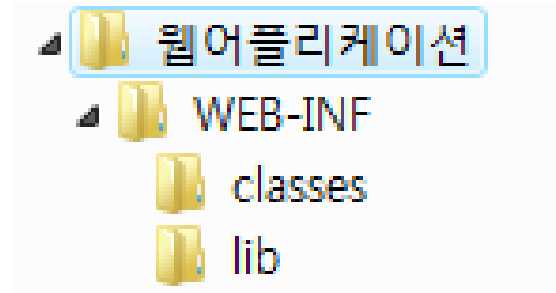
■ WAR 파일 생성

- Export → WAR file



■ 웹 어플리케이션 디렉터리 구성

- 일반적 구성



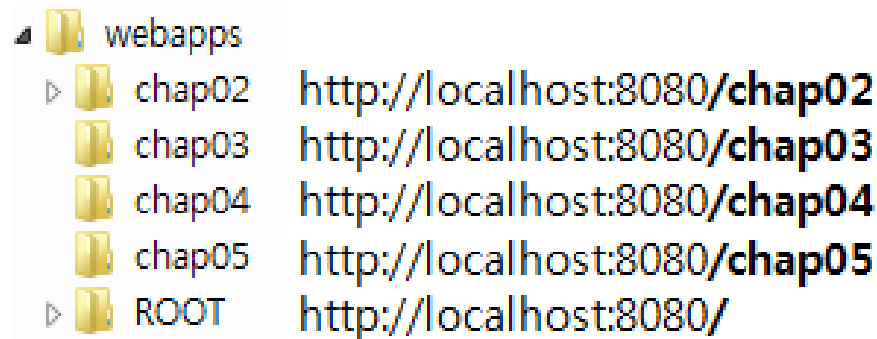
- 디렉터리 설명

- WEB-INF - web.xml 파일이 위치
- WEB-INF\classes - 웹 어플리케이션에서 사용하는 클래스 파일이 위치
- WEB-INF\lib - 웹 어플리케이션에서 사용하는 jar 파일이 위치

- JSP 2.1(서블릿2.5)부터는 web.xml 파일 선택적 필요

■ 웹 어플리케이션 디렉터리와 URL 구성

- 웹 어플리케이션 디렉터리 이름 → 컨텍스트 경로
- 컨텍스트 경로 → URL
 - 예, 컨텍스트 경로가 /chap02인 경우
URL은 `http://host:port/chap02`로 매핑
- 톰캣의 경우 [톰캣]webapps 디렉터리에 어플리케이션 디렉터리 위치



A screenshot of a file explorer showing the contents of the 'webapps' directory. The directory is expanded, showing subdirectories 'chap02', 'chap03', 'chap04', 'chap05', and 'ROOT'. Each subdirectory is accompanied by its corresponding URL mapping.

webapps	
chap02	<code>http://localhost:8080/chap02</code>
chap03	<code>http://localhost:8080/chap03</code>
chap04	<code>http://localhost:8080/chap04</code>
chap05	<code>http://localhost:8080/chap05</code>
ROOT	<code>http://localhost:8080/</code>

■ 웹 어플리케이션 디렉터리와 URL 구성

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<html>
<head> <title> 웹 어플리케이션 경로 구하기 </title> </head>
<body>
```

웹 어플리케이션 컨텍스트 경로: "<%= request.getContextPath() %>"

```
</body>
</html>
```


■ 웹 어플리케이션 배포

- 보통 두 가지 방식으로 배포
 - 대상 디렉터리에 직접 복사
 - war 파일로 묶어서 배포
 - 톰캣의 경우 [톰캣]webapps에 war 파일 복사
 - war 파일의 이름이 보통 컨텍스트 경로가 됨
- 컨테이너에 따라 배포 툴을 제공하기도 함